BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PTO/SB/21 (08-03)
Approved for use through 08/30/2003. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/605,520
Filing Date	10/06/2003
First Named Inventor	Han-Wen Hsu
Art Unit	
Examiner Name	
Attorney Docket Number	MTKP0040USA

Total Number of Pages in This Submission: 3

ENCLOSURES *(Check all that apply)*

- ☑ Fee Transmittal Form
- ☐ Fee Attached
- ☐ Amendment/Reply
- ☐ After Final
- ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☑ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
- ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance communication to Technology Center (TC)
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☐ Other Enclosure(s) (please Identify below):

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Winston Hsu, Reg. No.: 41,526
Signature	Winston Hsu
Date	12/8/2003

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name			
Signature		Date	

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/17 (10-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 0.00

Complete if Known	
Application Number	10/605,520
Filing Date	10/06/2003
First Named Inventor	Han-Wen Hsu
Examiner Name	
Art Unit	
Attorney Docket No.	MTKP0040USA

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☑ Deposit Account:

Deposit Account Number: 50-0801

Deposit Account Name: North America International Patent Office

The Director is authorized to: *(check all that apply)*

☑ Charge fee(s) indicated below ☑ Credit any overpayments

☑ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) ($) 0.00

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

	Extra Claims		Fee from below	Fee Paid
Total Claims		-20** =		X =
Independent Claims		- 3** =		X =
Multiple Dependent				=

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0.00

***or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 0.00

SUBMITTED BY

(Complete *(if applicable)*)

Name (Print/Type)	Winston Hsu	*Registration No. (Attorney/Agent)*	41,526	*Telephone*	886289237350
Signature	Winston Hsu			*Date*	12/19/2003

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/02B (11-00)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

DECLARATION — Supplemental Priority Data Sheet

Additional foreign applications:

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached?	
				YES	NO
092105022	Taiwan R.O.C	03/07/2003	☐	☑	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐

Burden Hour Statement: This form is estimated to take 21 minutes to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

中華民國經濟部智慧財產局

INTELLECTUAL PROPERTY OFFICE
MINISTRY OF ECONOMIC AFFAIRS
REPUBLIC OF CHINA

茲證明所附文件，係本局存檔中原申請案的副本，正確無訛，
其申請資料如下 ：

This is to certify that annexed is a true copy from the records of this office of the application as originally filed which is identified hereunder :

申　　請　　日 ： 西元 2003 年 03 月 07 日
Application Date

申　請　案　號 ： 092105022
Application No.

申　　請　　人 ： 聯發科技股份有限公司
Applicant(s)

局　　長
Director General

蔡練生

發文日期： 西元 2003 年 4 月 1 日
Issue Date

發文字號：
Serial No. 09220318130

申請日期：	IPC分類
申請案號：	

(以上各欄由本局填註)

發明專利說明書

一、發明名稱	中 文	統一處理多個子程序執行回應之韌體架構方法及相關裝置
	英 文	Firmware Structuring Method And Related Apparatus For Unifying Handling Of Execution Responses Of Subroutines
二、發明人(共2人)	姓 名(中文)	1. 許漢文 2. 蔡明憲
	姓 名(英文)	1. Hsu, Han-Wen 2. Tsai, Ming-Hsien
	國 籍(中英文)	1. 中華民國 TW 2. 中華民國 TW
	住居所(中 文)	1. 新竹市關東路二四七號四樓 2. 高雄市前金區六合二路一四0巷二十八號
	住居所(英 文)	1. 4F, No. 247, Kuan-Tung Rd., Hsin-Chu City, Taiwan, R.O.C. 2. No. 28, Lane 140, Liu-Ho II Rd., Chien-Ching, Kao-Hsiung City, Taiwan, R.O.C.
三、申請人(共1人)	名稱或姓 名(中文)	1. 聯發科技股份有限公司
	名稱或姓 名(英文)	1. MediaTek Inc.
	國 籍(中英文)	1. 中華民國 TW
	住居所(營業所)(中 文)	1. 新竹市新竹科學工業園區創新一路13號1F (本地址與前向貴局申請者相同)
	住居所(營業所)(英 文)	1. 1F, No. 13, Innovation Road 1, Science-Based Industrial Park, Hsin-Chu City, Taiwan, R.O.C.
	代表人(中文)	1. 蔡明介
	代表人(英文)	1. Tsai, Ming-Kai

四、中文發明摘要　（發明名稱：統一處理多個子程序執行回應之韌體架構方法及相關裝置）

本發明係提供一種裝置控制之韌體程式架構方法及相關裝置。該韌體程式係由一處理器來執行，以控制一硬體電路之運作。其中該韌體程式中設有複數個子程序以分別定義該硬體電路之不同運作，而該複數個子程序被區分為數個不同的階層，階層較低的子程序用來定義該硬體電路較簡單的運作，而階層較高的子程序則呼叫複數個較低階層之子程序以定義出該硬體電路較複雜的運作；而各個低階子程序會將該硬體電路對應運作之結果記錄於一錯誤記錄碼。其中當該硬體電路進行某運作而未達到該運作之預期結果時，該硬體電路應進行一對應的回復運作。為了控制該硬體電路進行必要的回復運作，該韌體程式中設有一錯誤處理子程式，用來在一高階子程序完成後，依據錯誤記錄碼使該硬體電路進行對應該高階子程序中各低階子程序之回復運作。

六、英文發明摘要　（發明名稱：Firmware Structuring Method And Related Apparatus For Unifying Handling Of Execution Responses Of Subroutines ）

A firmware code structuring method and related apparatus. The firmware code is executed b, a processor to control operation of a hardware circuit. Wherein the firmware code includes a plurality of subroutines to define various operations of the hardware circuit, and the subroutines are grouped to several different levels. A subroutine of lower level defines a

四、中文發明摘要　(發明名稱：統一處理多個子程序執行回應之韌體架構方法及相關裝置)

伍、(一)、本案代表圖為：圖五F

(二)、本案代表圖之元件代表符號簡單說明：

46	韌體程式碼	F1-F17	箭頭
EH	錯誤處理子程式	IF	介面程式組
SR	伺服程式組		

A01-A02_1、B01_2-B07_2、C01_3-C03_3、D01_4-D02_4、E01_5-E03_5　子程序

六、英文發明摘要　(發明名稱：Firmware Structuring Method And Related Apparatus For Unifying Handling Of Execution Responses Of Subroutines)

simpler operation of the hardware circuit, and a higher level subroutine calls a plurality of lower level subroutines to define more complicated operations of the hardware circuit. When the lower level subroutines are executed, they dump results of corresponding operations in an error code. Wherein if certain operations performed do not achieve expected results, a

四、中文發明摘要　（發明名稱：統一處理多個子程序執行回應之韌體架構方法及相關裝置）

六、英文發明摘要　（發明名稱：Firmware Structuring Method And Related Apparatus For Unifying Handling Of Execution Responses Of Subroutines ）

corresponding recovery operation should be performed by the hardware circuit. To control t..e hardware to perform required recovery operations, a error-handler is executed to make the hardware circuit perform recovery operations corresponding to lower level subroutines called in a higher level subroutine according to the error code after the higher level subroutine is

四、中文發明摘要　（發明名稱：統一處理多個子程序執行回應之韌體架構方法及相關裝置）

六、英文發明摘要　（發明名稱：Firmware Structuring Method And Related Apparatus For Unifying Handling Of Execution Responses Of Subroutines ）

finished.

一、本案已向

國家(地區)申請專利	申請日期	案號	主張專利法第二十四條第一項優先權
無			

二、□主張專利法第二十五條之一第一項優先權:

申請案號: 無

日期:

三、主張本案係符合專利法第二十條第一項□第一款但書或□第二款但書規定之期間

日期:

四、□有關微生物已寄存於國外:

寄存國家:

寄存機構: 無

寄存日期:

寄存號碼:

□有關微生物已寄存於國內(本局所指定之寄存機構):

寄存機構:

寄存日期: 無

寄存號碼:

□熟習該項技術者易於獲得,不須寄存。

五、發明說明（1）

發明所屬之技術領域

本發明提供一種韌體架構方法及相關裝置，尤指一種以分層管理、錯誤回復集中處理來減少韌體程式複雜度的方法與相關裝置。

先前技術：

在現代化的資訊社會中，資訊、影像、數據資料都是以電子形式的訊號來傳輸、儲存及處理的，而各種各樣至電腦，也電子裝置中，多半設有一處理器（或稱微處理器）統合操控電子裝置的各種運作；而在功能繁複、多的電子系統中，由於操控的程序比較複雜，就要用特定程式碼來執行程序進行的相關步驟及過程；而韌體程式碼；而韌體（像是能式碼，即被稱之為韌體程式碼，就能實現電子裝置的不同功快閃記憶體）中，方便處理器讀取、執行。另外，像在電腦系統這種功能更為複雜多樣的電子系統中，電腦的各種周邊裝置本身就具有自己的處理器及對應的韌體程式碼；電腦的主機本身僅需發出高階的控制指令至周邊裝置的處理器，該周邊裝置的處理器就會執行該周邊裝

置本身的韌體程式碼，控制其周邊裝置的實際運作。舉例來說，電腦系統中的光碟機本身就具有其處理器及對應的快閃記憶體，以儲存光碟機本身的韌體程式碼。當電腦的主機要讀取一光碟片上的資料時，僅需向光碟機指定該筆資料在光碟片上的位址，光碟機的處理器就會執行本身的韌體程式碼，協調光碟機中諸如主軸馬達(spindle)、雷射讀取頭(pick-up head)等等機電元件之運作（像是主軸馬達要達到特定的轉速，讀取頭要移動、鎖定至特定的位置，接收光碟片上反射的雷射等等），實際達成主機的要求。

請參考圖一。圖一為一典型周邊裝置12配合一主機10運作時之功能方塊圖。周邊裝置12為一典型的電子裝置，其中即設有一處理器16，用來主控周邊裝置12的運作。除此之外，周邊裝置12中還設有一揮發性的緩衝記憶體22（像是隨機存取記憶體）、一非揮發性的儲存記憶體24（像是快閃記憶體）以及一用來實現周邊裝置12功能的硬體電路18。周邊裝置12通常電連於主機10，以接收主機10的操控指令而運作。主機10可以為一電腦主機，其中設有一中央處理器14A、一北橋電路14B、一南橋電路14C、一顯示卡14E、一顯示器14F、一揮發性的記憶體14D（像是隨機存取記憶體）。中央處理器14A用來主控主機10的運作，記憶體14D用來暫存中央處理器14A運作時必需的資料、程式，顯示卡14E用來處理影像訊

號，以將主機10運作的情形於顯示器14F顯示為影像畫面。北橋電路14B用來控制顯示卡14E、記憶體14D與中央處理器14A之間的資料傳輸；南橋電路14C則透過北橋電路14B電連於中央處理器14A，與主機10配合運作的周邊裝置12即透過與南橋電路14C間的電連（像是透過一IDE匯流排的電連）而得以和主機10交換指令、資料。

在周邊裝置12中，除了主控周邊裝置12運作的處理器16外，緩衝記憶體22用來暫存周邊裝置12運作期間必需的資料；而硬體電路18中設有一編解碼器20A、一訊號處里器20B及一伺服模組20C。由於主機10、周邊裝置12間往來傳輸之指令、資料需要符合一定的格式、協定(protocol)，故編解碼器20A會將主機10傳至周邊裝置12的指令、資料解碼，再由處理器16依據解碼後的指令、訊號來操控周邊裝置12；同理，由周邊裝置12回傳至主機10的資料也會由編解碼器20A加以適當地編碼，以符合主機10、周邊裝置12間資料交換的格式及協定。伺服模組20C則受控於處理器16，以實際執行周邊裝置12的主要功能；其所存取得之資料、訊號會由訊號處理器20B加以訊號處理。舉例來說，若周邊裝置12為一用來存取光碟片資料之光碟機，則伺服模組20C中會設有一主軸馬達28A、一讀取頭28B等等機電元件。主軸馬達28A用來帶動光碟片28C轉動；讀取頭28B則可沿一滑軌28D滑動，以存取光碟片上不同軌道(track)上的資料。伺服模組20C由

光碟片 28C讀到的資料，會經過訊號處理器 20B的訊號處理，再經由處理器 16的安排儲存入緩衝記憶體 22中，讓主機 10可透過編解碼器 20A取得緩衝記憶體 22中的這些資料；這樣主機 10就能透過光碟機之周邊裝置 12讀取到光碟片 28C上的資料。同理，若主機 10要將資料寫入至光碟片 28C上，則會將資料透過編解碼器 20A傳輸至緩衝記憶體 22，再由處理器 16的控制，利用伺服模組 20C將這些暫存於緩衝記憶體 22的資料寫入至光碟片 28C。

如前所述，周邊裝置 12的處理器 16會依據記錄於程式中的操控程序控制周邊裝置 12之運作，而儲存於非揮發性儲存記憶體 24中的韌體程式碼 26，就是用來記錄這些操控程序。處理器 16執行韌體程式碼 26，就可操控周邊裝置 12進行各種不同的運作。

請參考圖二 A（並一併參考圖一）。圖二 A為習知技術中，韌體程式碼 26典型程式結構之示意圖。韌體程式碼 26的程式可分為兩大類，一為介面程式組 IF0、一為伺服程式組 SR0。伺服程式 SR0中包括有多個子程序（subroutine，圖二 A中繪出子程序 R1、R2A-R2B、R3A-R3B、R4A-R4B、R5A-R5C、R6A-R6B及 R7A-R7B做為代表），各個子程序用來控制硬體電路 18中的各電路進行某種特定的運作。舉例來說，某個子程序可控制伺服模組 20C中的讀取頭 28B由滑軌 28D上的某個位置移到另一位

置，另一個子程序則可控制讀取頭28B調整其雷射功率的大小，等等。介面程式組IF0中也可包含多個子程序，介面程式組IF0的主要功能則是用來根據主機10之指令，呼叫伺服程式組SR0的對應子程序，以操控硬體電路18達成主機10指定的功能。伺服程式組SR0執行後的結果，也會再由介面程式組IF0之執行而回報至主機10。舉例來說，若主機10要求周邊裝置12讀取光碟片上的某段資料，介面程式組IF0就會呼叫伺服程式組SR0中定義有資料讀取操控程序之子程序；處理器16執行此子程序，就能操控硬體電路18，完成資料讀取的工作。若子程序順利完成，可發出工作完成之訊息（像是將訊息存在一變數中），再由介面程式組IF0依此訊息控制周邊裝置12對主機10回傳相關的訊號，知會主機10其工作的結果。

一般來說，由於現代電子裝置的功能日趨多元，裝置內部的操控程序、韌體程式碼也越來越複雜。為了縮短韌體程式碼的研發時程，資訊業者多半是由多個韌體工程師協同運作，各個韌體工程師負責撰寫一部份的子程序，再統合至韌體程式碼中。如熟知技術者所知，子程序可將特定的操控程序模組化，方便重複利用。舉例來說，在光碟機中，不論是開機後的初始化動作、要讀取光碟片上特定部分的資料，或是要將資料寫入至光碟片上的某一部份時，可能都要控制讀取頭28B由滑軌28D上的某一位置移動到另一位置，而此控制讀取頭28B移動

的程式碼就可模組化於一子程序中。而在操控光碟機進行開機初始化動作、讀取資料、寫入資料的不同子程序中，就能呼叫（call)這個控制讀取頭移動的子程序，以操控讀取頭28B移動。這樣一來，開發各個子程序的韌體工程師就不必在開機初始化動作、讀取資料、寫入資料的各個子程序中重複編寫控制讀取頭移動的程式碼。

雖然子程序的運用能在不同的操控程序中共用相同的子程序，便利韌體程式碼的開發，但在習知技術中，卻缺乏對子程式呼叫的控制管理。尤其是當各子程式是由不同的韌體工程師開發時，開發某一子程序A的韌體工程師可能會呼叫其他韌體工程師所開發的子程序B，但子程序A的韌體工程師並不一定清楚子程序B中的運作細節，可能子程序B中又要呼叫另一子程序C；子程序C中又另外呼叫了子程式D，等等。這樣一來，各子程序間互相呼叫的運作流程就會越來越複雜而難以掌控。關於此情形，請再參考圖二A。圖二A中的各箭號就用來表示圖二A中各個子程序互相呼叫的流程。舉例來說，根據介面程式組IF0的呼叫，處理器16要執行伺服程式組SR0中的子程序R1（箭頭A1代表子程序R1開始被執行）。在子程序R1中先後呼叫了子程序R2A及R2B，故箭頭A2代表子程序R2A開始被執行，子程序R2A執行完後，接下來的箭頭A3代表子程序R2B被執行。子程序R2B本身又呼叫了子程序R3A、R3B，故在子程序R2B完成前，要先如箭頭A4所指

示，先執行子程序R3A，再依箭頭A5，執行子程序R3B。子程序R3B中又先後呼叫了子程序R4A、R4B，故程式又如箭頭A6、A7所指示，在執行子程式R4A後，開始執行子程序R4B。子程序R4B中又呼叫了子程序R5A、R5B及R5C，故在子程序R4B完成前，又要如箭頭A8、A9及A10，依序完成子程序R5A及R5B。在執行子程序R5C時，因為子程序R5C本身又先後呼叫了子程序R6A、R6B，故要如箭頭A11指示的順序先執行子程序R6A。而子程序R6A中又先後呼叫了子程序R7A、R7B，故程式執行的順序就如箭頭A12、A13所示，先後執行了子程序R7A、R7B後，才能如箭頭A14所示般繼續執行子程序R6A，完成子程序R6A後再繼續如箭頭A15所示執行子程序R6B。在執行完子程序R5C中呼叫的兩個子程序R6A、R6B後，才能再如箭頭A16之指示般繼續子程序R5C。子程序R5C結束後依箭頭A17之順序回到子程序R4B，繼續執行子程序R4B在呼叫子程序R5A至R5C後剩下的操控程序。子程序R4B完成後，如箭頭A18的指示回到子程序R3B之執行。子程序R3B完成後，呼叫子程序R3A、R3B的子程序R2B才能繼續執行（如箭頭A19）。等子程序R2B執行完後，執行的流程才會如箭頭A20所指引，繼續回到子程序R1的執行。等子程序R1完成後，就能如箭頭A21之指示，將子程序R1執行之結果回傳至介面程式組IF0。

由上述描述可知，子程序之呼叫若無適當的管理，

很有可能會使執行流程變得複雜，難以追蹤。舉例來說，開發子程序R1的韌體工程師可能只因子程序R2B之功能符合其操控程序中的某一步驟，故在子程序R1中呼叫了子程序R2B，卻忽略了子程序R2B中還會連串地呼叫子程序R3A、R3B，而子程序R3B又要呼叫其他的子程序。這樣一來，開發子程序R1的韌體工程師就不能掌控子程序R1執行時的真正流程。若開發子程序R1的韌體工程師還要自行追蹤子程序R2A的執行流程，又失去了子程序將程式碼模組化而方便運用的意義。

另外，複雜、冗長的執行流程也會大量耗用周邊裝置12本身的處理器資源。舉例來說，如熟知技藝者所知，當子程序R1在執行期間要執行其所呼叫的子程序R2A時（如箭頭A2），子程序R1中各變數之值要暫存入緩衝記憶體22中配置的堆疊（stack），不能釋放。接下來在子程序R2B執行時，因其呼叫了子程序R3A、R3B，故又要先將子程序R2B中的各變數值暫存入堆疊中，不能釋放。在繼續執行子程序R3A、R3B時，又要在呼叫子程序R4A、R4B之前，先將子程序R3B執行期間的相關變數值暫存入堆疊中，不能釋放。換句話說，在後續的執行流程中，當某一子程序A呼叫的子程序B未完成前，子程序A的變數都必需要暫存於堆疊中，不能釋放。可想而知，若是執行流程中有連串的子程序呼叫（也就是在某一子程序未執行完成就要執行另一子程序），暫存在堆疊中的變數

值就越來越多，大量耗費緩衝記憶體22中的記憶空間。

除此之外，冗長、繁複的子程序連串呼叫也會使整體程式的除錯(debug)更為困難。一旦程式錯誤(bug)發生，韌體工程師勢將檢查執行流程中的各個子程序，才能找出程式錯誤的癥結所在。另外，在針對某一子程序除錯時，由於韌體工程師無法獲悉執行流程中前後子程序執行之情形，也增加了除錯時的困難。舉例來說，當韌體工程師在檢查子程序R4B時，因為無從得知子程序R3A、R3B以及子程序R5A至R5C的執行情形，將難以判斷子程序R4B是否運作正常。因為子程序R4B若有程式錯誤，可能是子程序R3A本身出錯而影響子程序R4B的執行，或是因子程序R5A至R5C的程式錯誤而造成子程序R4B的整體錯誤。

除了上述缺點之外，連串之子程式呼叫也可能造成周邊裝置12運作錯誤(error)難以正確回復。由於周邊裝置12中的韌體程式碼26是用來整合周邊裝置12中各機電元件運作情形的，若韌體程式碼26執行期間有某機電元件失常（或使用者突然改變對周邊裝置的控制），就會打斷韌體程式碼26正常的執行流程，造成周邊裝置12的運作錯誤；此時就要進行對應的錯誤回復(error recovery)運作，以恢復周邊裝置12的正常運作。舉例來說，若光碟機在存取光碟片資料期間遭到突然的撞擊震

動而使得讀取頭28B的位置改變，光碟機可能就要進行重新鎖軌（或另行尋軌）之回復運作，以使光碟機回復正常的資料存取狀態。而韌體程式碼26中，當然也要包含錯誤回復之操控流程，以控制周邊裝置12在運行錯誤發生時，進行回復運作。關於此情形，請參考圖二B。圖二B與圖二A相同，都用來顯示習知技術中韌體程式碼26程式結構之示意圖。不過，圖二B中更進一步地繪出習知技術錯誤回復機制的實施方法。如圖二B所示，在處理器16執行子程序R3B而控制周邊裝置12運作時，若周邊裝置12發生運作的錯誤，就會在步驟R3s的邏輯控制中進行至子程序R8A、R8B，以操控周邊裝置12進行回復運作；等子程序R8A、R8B執行無誤後，子程序R1的執行流程才會繼續如箭頭A19之指示回到子程序R2B之執行。

不過，在習知技術中，由於缺乏對錯誤回復之管理，也容易在子程式連串呼叫的複雜執行流程中，造成不正確的錯誤回復。舉例來說，如圖二B所示，子程式碼R5A本身也包括了錯誤回復的機制；若周邊裝置12執行子程式碼R5A時發生運作錯誤，就會由步驟R5s的邏輯控制進行至子程序R9，以進行錯誤回復。然而，開發子程序R?B的韌體工程師在子程序R3B中呼叫了子程序R4A、R4B後，可能就忽略了子程序R4B呼叫的子程序R5A中，已包含了對子程序R5A錯誤回復的機制，使得在負責子程序R3B錯誤回復的子程序R8A、R8B中又再度重複與子程序

R5A錯誤回復相關的操控流程。重複進行的錯誤回復可能會導致執行流程的遲滯，甚至導致周邊裝置12額外的運作錯誤。同樣地，由於開發子程序R3B的韌體工程師未察覺子程序R4B最終會呼叫到子程序R6A，也會使得錯誤回復之子程序R8A、R8B中未包含對應子程序R6A運作錯誤的回復運作。如此一來，一旦子程序R6A執行期間周邊裝置12發生運作錯誤，周邊裝置12也無法進行正確的錯誤回復。換句話說，在習知技術下，因為在子程序連串呼叫的執行流程中缺乏對錯誤回復的管理，有可能會使運作錯誤得不到適當的錯誤回復，或進行過多、冗餘的錯誤回復，或是沒有進行任何錯誤回復。

總結來說，因為在習知技術在韌體程式碼中缺乏對子程序呼叫及錯誤回復之有效管理，使得習知技術中的韌體程式碼架構複雜，難以追蹤、除錯，可讀性較低，而且執行時會大量消耗周邊裝置中的處理器資源，還容易因錯誤回復缺乏管理而影響周邊裝置的正常運作。

發明內容：

因此，本發明之主要目的在於提供一種對韌體程式碼新的架構方法及相關裝置，能有效管理韌體程式碼中各子程序相互呼叫的秩序，並統一對各種運作錯誤進行錯誤回復，以增強對錯誤回復之管理，克服習知技術的

缺點。

在習知技術中，由於對韌體程式碼中的各個子程序相互呼叫並沒有一定的管理規範，導致習知技術下的韌體程式碼會形成複雜的連串呼叫，不僅程式的可讀性甚低，難以追縱、除錯，且在執行時會消耗大量的處理器資源。同樣地，習知技術也缺乏對各子程序錯誤回復機制的管理，會導致周邊裝置無法對運作錯誤採取有效的錯誤回復、或是重複錯誤回復運作，甚至是無法採取有效的錯誤回復。

在本發明中，則提出兩基礎原則以管理一周邊裝置的架構之韌體程式碼中各子程序相互呼叫、進行錯誤回復的架構。首先是依照各子程序對應運作之複雜程度將各子程序分類為不同階層的子程序。階層較低、較後的子程序用來定義周邊裝置較單純的運作。階層較高、較前的子程序則可呼叫多個低階子程序以定義出對周邊裝置較複雜的操控程序。由於各高階子程序是以呼叫不同低階子程序的方式來組合出各種操控程序，故可有效管理各子程序間互相呼叫的秩序，並易於控制韌體程式碼執行流程中各子程序連串呼叫的情形。

其次，本發明另設有一錯誤處理子程序，用來統合各子程序對應之錯誤回復。當周邊裝置在執行各子程序

時，會依照子程序中的程式將周邊裝置本身進行實際運作之結果記錄於一錯誤記錄碼；等到高階子程序完成後，周邊裝置才會執行錯誤處理子程序，依照錯誤記錄碼中記錄的內容來操控周邊裝置進行對應的錯誤回復運作。換句話說，各子程序本身並不會進行錯誤回復運作，而是另行於錯誤處理子程序中統一定義對應各子程序的錯誤回復運作。這樣一來，就可避免在子程序連串呼叫的過程中啟動不正確的錯誤回復。

實施方式：

請參考圖三。圖三為本發明中一周邊裝置32配合一主機30運作之功能方塊示意圖。主機30可以是一電腦系統的主機，其可設有一中央處理器34A、一北橋電路34B、一南橋電路34C、一記憶體34D、一顯示卡34E及一顯示器34F。周邊裝置32則可以是電腦系統中用來擴充主機功能的周邊裝置，像是光碟機、硬碟機等等；周邊裝置32中設有一用來主控周邊裝置32運作之處理器36、一用來在周邊裝置32運作期間暫存資料的揮發性緩衝記憶體42(像是一隨機存取記憶體)、一非揮發性的儲存記憶體45以及一用來實現周邊裝置32功能之硬體電路38。在主機30中，中央處理器34A用來主控主機30的運作，揮發性的記憶體34D(像是隨機存取記憶體)則用來暫存主機30運作期間所必需的資料、數據；顯示卡34E用來處理

影像資料，以將主機30運作的情形在顯示器34F上顯示為影像畫面。北橋電路34B用來控制顯示卡34E、記憶體34D及中央處理器34A之間的資料往來傳輸。透過電連於北橋電路34B之南橋電路34C，主機30和周邊裝置32就能交換資料、指令。在周邊裝置32中，硬體電路40C中設有一編解碼器40A、一訊號處理器40B、一伺服模組40C。由主機30傳至周邊裝置32的指令資料會由編解碼器40A加以解碼，再由處理器36接收、處理。伺服模組40C存取的資料、訊號則可由訊號處理模組40B進行訊號處理。而伺服模組40C中設有用來實現周邊裝置32功能之機電元件；舉例卡說，若周邊裝置32為一光碟機，則伺服模組40C中可包括有一用來帶動光碟片48C轉動的主軸馬達48A、一可沿滑軌48D移動的讀取頭48B，等等。儲存記憶體45可以為一快閃記憶體，處理器36即是執行儲存記憶體45中儲存的韌體程式碼46，以在接受主機30之控制指令後，操控硬體電路38進行周邊裝置32的預設功能。

本發明之主要目的在於提出一種新的韌體程式碼架構模式。本發明提出兩種韌體程式碼架構之原則。首先，依本發明之原則，韌體程式碼中不同的子程序可區分為不同的階層。階層較低、較後階的子程序用來定義周邊裝置32較基本、較簡單、功能較為單一的運作。階層較高，較前階的子程序用來呼叫較低階的子程序，以組合出較複雜、功能較為完整的操控程序，或是較複雜

的邏輯。此處所謂的較複雜之邏輯，是指當某邏輯條件為真時，要進行一連串較複雜的操控程序；當該邏輯條件為偽時，又要進行另一連串較複雜的操控程序。在本發明中，這樣的複雜邏輯，就可用較高階的子程序來整合。本發明揭露的另一原則是，統一以一錯誤處理子程序來管理、執行其他各個子程序所需的錯誤回復。換句話說，當周邊裝置32在執行一高階子程序所呼叫的各個低階子程序時，即使發生運作上的錯誤，也不會立即進行對應的錯誤回復運作，而是將運作錯誤發生的情形記錄於一錯誤記錄碼。等高階子程序結束後，周邊裝置32才會執行錯誤處理子程序，根據錯誤記錄碼中記錄的錯誤發生情形而呼叫對應的子程序，以進行錯誤回復運作。而用來定義錯誤回復運作的子程序，也可稱之為回復子程序。以錯誤處理子程序來統一管理各子程式對應的回復運作，就能讓韌體程式碼46正常運作之執行流程與錯誤回復進行之流程兩者獨立，使正常運作執行流程的運作以及錯誤回復機制都變得更單純，對韌體工程師來說，程式之控管、追蹤、除錯和程式之可讀性都可大大的提高。

為進一步說明本發明實施的情況，請參考圖四。圖四為本發明中韌體程式碼46架構之示意圖。如前所述，韌體程式碼46中的子程序可分為兩大類，一為介面程式組IF，另一組為伺服程式組SR。伺服程式組SR中的各個

子程序用來定義周邊裝置32各種不同的操控程序，介面程式組IF中的子程序則根據主機30之控制指令呼叫伺服程式組中對應的程式，使周邊裝置32進行主機30控制指令所要求的運作。同理，周邊裝置32也可執行介面程式組IF中的子程序，將周邊裝置32執行伺服程式組SR中各子程序之結果編成適當格式的資料，回報至主機30。為了說明上的方便，在不妨礙本發明技術揭露的情形下，以下將假設本發明係實施於伺服程式組SR中的各個子程序。

如前所述，本發明係將各子程序歸類為不同的階層。在圖四中的實施例，即將伺服程式組SR中的各個子程序分組為五個不同的階層，最高階（最前階）的子程序有子程序A01_1、A02_1等等（圖四中繪出兩個做為代表），其次的第二階子程序中則有子程序B01_2、B02_2、B03_2、B04_2到B07_2等等，第三階的子程序包括有子程序C01_3、C02_3、C03_3等等，第四階子程序則有子程序D01_4、D02_4等等，最低階（最後階）的子程序則包括有子程序E01_5到E03_5等等；另外，子程序EH則是錯誤處理子程序。最低階的子程序E01_5、E02_5等等子程序用來定義周邊裝置32較基本、較單純（沒有複雜邏輯）的運作，甚至只是用來設定周邊裝置32運作期間必要的參數。較高階的子程序則用來呼叫不同的低階子程序，以組合出較為複雜的操控程序及較為複雜的邏

輯。像是第四階的子程序D01_4、D02_4就可以呼叫各個不同的第五階子程序，來組合出不同的操控程序。同理，第三階的子程序C01_3、C02_3及C03_3可以呼叫第四階、第五階的子程序，組合出比第四階子程序更複雜的操控程序。第二階的子程序B01_2等等則能進一步地呼叫第三階、第四階、第五階的各個子程序，組合出更複雜、功能更完整的操控程序。最後，到最高階的第一階子程序，則能呼叫第二階至第五階的各種子程序，共同組合出功能最完整、也最複雜的操控程序或最複雜的邏輯。

在本發明之較佳實施例中，除了最低階的子程序可以互相呼叫外，其他階的子程序中，屬於同一階層的子程序均不互相呼叫。舉例來說，圖四中最低階的子程序E01_5可以呼叫子程序E02_5及E03_5；也就是說，子程序E01_5在執行期間可以先執行子程序E02_5及E03_5，等子程序E03_5完成後，再執行子程序E01_5中剩下的操控程序。不過，較高階的各個子程序中，屬於同一階層的各個子程序皆不互相呼叫。舉例來說，在第三階的子程序C01_3、C02_3及C03_3不會互相呼叫，也就是在（舉例而言）子程序C02_3執行期間，執行流程不會去執行其他的第三階子程序，直到子程序C02_3執行完畢，執行流程才有可能去執行其他的第三階子程序。此外，在本發明中，較低階的子程序也不呼叫較高階的子程序。舉例來

說，若周邊裝置32有某一操控程序需要依序執行子程序E01_5、E02_5以及D01_4，此時應以定義一第三階子程序來整合呼叫這三個較低階的子程序，而不是另行定義第四階、第五階之子程序來呼叫這三個本屬第四階、第五階之子程序E01_5、E02_5及D01_4。

藉由本發明上述呼叫秩序之安排，韌體程式碼46就能避免複雜的子程序連串呼叫。在最低階的子程序（也就是圖四中的第五階子程序）中，由於最低階的子程序僅用來定義周邊裝置32最基本、最單純、功能也較為單一的運作，故最低階子程序間的相互呼叫，並不會形成複雜、難以追蹤的連串呼叫，也不會大量消耗處理器36的處理器資源。較高階的子程序因為有較為複雜的功能及邏輯，故本發明由高階子程序來呼叫低階子程序，並限制同一階層子程序之互相呼叫，就能有效避免子程序間複雜無秩序之連串呼叫。換句話說，依本發明之原則來管理子程序間的呼叫後，就能有效控制子程序間相互連串呼叫的執行流程。舉例來說，在圖四中的五階實施例中，第一階的子程序可呼叫第二階的子程序、第二階的子程序會呼叫第四階的子程序、第三階的子程序會呼叫第五階的子程序，但子程序連串呼叫的次數大致上就會限制於四次，不會再增加子程序連串呼叫的複雜度（如前所述，第五階子程序的相互呼叫幾乎不會增加子程序連串呼叫

的複雜度）；這就是因為實施了本發明所揭露之高階至低階的呼叫秩序，以及同一階層子程序不互相呼叫的原則。試以一反例來思考，若第一階子程序呼叫了一第二階子程序B01_2、而子程序B01_2又呼叫了同一階子程序B02_2，則子程序連串呼叫的次數就會至少增加一次。若某一子程序E02_5又呼叫了一高階的子程序，像是子程序B02_2，那麼子程序連串呼叫的次數及複雜度又會大幅增加，因為子程序E02_5呼叫了子程序B02_2後，子程序B02_2又會呼叫第三階、第四階到第五階的子程序。無限制、無秩序地讓各個子程序互相呼叫，正是習知技術中無去控制子程序連串呼叫複雜度之原因。相較之下，利用本發明所揭露之原則來管理子程式相互間的呼叫後，就能有效控制子程序連串呼叫的次數及複雜度，也不影響利用子程式來整合各種操控程序、減少重複程式碼之目的。

除了階層化的呼叫秩序外，本發明還以一錯誤處理子程式EH（見圖四）來整合不同操控程序運作錯誤所對應的錯誤回復。配合錯誤處理子程式EH之運作，各子程序會將運作錯誤之情形記錄於一作為錯誤記錄碼之全域變數（global variable），讓各個子程序皆可以存取此一錯誤記錄碼之值。為進一步說明此情形，請參考圖五A至五D（並一併參考圖四）。圖五A至五D列示的是子程序A01_1、B01_2及C01_3、D01_4之程式碼的示意例。圖五A

中也定義了後續說明所使用的一些巨集（像是巨集ChkStatus等等）。在此處（及後續）所列示的程式碼，皆是以C程式語言的格式來示意各子程序的內容，但在實際實施時，本發明中之各子程序當然可以使用其他的程式語言來形成。另外，在不妨礙本發明技術揭露的情形下，詳細的程式碼（像是常數、變數、部分函式之宣告、定義等等）皆已省略；熟知技術人士應已可獲得足夠之技術揭露以實施本發明。如圖五A所示，在伺服程式組SR執行期間，是以一全域變數_bLevel來代表現行子程序之階層；就如圖四所顯示的，在第一階子程序執行期間，變數_bLevel之值應為1；同理，在執行第二階到第五階子程序時，變數_bLevel之值應分別為2到5。當子程序A01_1被執行前，變數_bLevel之值應被設為0。此外，陣列全域變數_bErrorCode即為錯誤記錄碼。

如圖五A所示，當第一階子程序A01_1被執行時，會將變數_bLevel加1，代表韌體程式碼46之執行流程正在進行一第一階子程序。接下來子程序A01_1會執行圖五A中的程式區段50A，依據一狀態變數_fgSelectB01_2之值進行邏輯判斷；若變數_fgSelectB01_2為真，則繼續呼叫第二階子程序B01_2；否則呼叫子程序B02_2。子程序B01_2、B02_2皆會回傳一位元組之值，分別代表周邊裝置32在執行子程序B01_2、B02_2後進行對應運作的結果。若子程序B01_2、B02_2回傳一常數READY之值，代表

周邊裝置32在進行子程序B01_2或B02_2時進行順利，並沒有發生運作錯誤。相反地，若周邊裝置32在執行子程序B01_2發生運作錯誤，子程序B01_2就不會回傳常數READY之值。如圖五A所示，若子程序B01_2回傳之值不是常數READY，子程序A01_1就會在錯誤記錄碼_bErrorCode的一個元素中（也就是_bErrorCode[1]），記錄下代表子程序B01_2運作錯誤的代號（即常數B01_Err之值），再將變數_bLevel重設為0，並中止子程序A01_1之執行，回傳常數值（!READY），代表子程序A01_1執行期間發生運作錯誤；此時錯誤記錄碼_bErrorCode[1]中即以常數B 1_Err記錄了此運作錯誤是在執行子程序B01_1時發生的。同理，若周邊裝置32在執行子程序B02_2的過程發生運作錯誤，子程序A01_1也會中止執行，回傳常數值(!READY)代表其發生了運作錯誤，並在錯誤記錄碼_bErrorCode[1]中記錄常數B02_Err之值，代表運作錯誤是在子程序B02_2執行期間發生的。

同理，接下來子程序A01_1會依序執行至程式區段50B、50C；以程式區段50B為例，子程序A01_1會在此程式區段中呼叫子程序B03_2，控制周邊裝置32進行子程序B^3_2對應之運作。若周邊裝置32在依據子程序B03_2進行對應運作卻發生運作錯誤時，子程序B03_2也就不會回傳常數READY之值；此時子程序A01_1就會依據子程序B03_2回傳之值判斷周邊裝置32已發生執行錯誤，並在錯

誤記錄碼 _bErrorCode[1]中記錄常數B03_Err之值，代表子程序B03_3執行期間發生了運作錯誤。接下來子程序A01_1就結束執行，回傳常數值(!READY)，代表子程序A01_1在執行期間因周邊裝置32之運作錯誤而中斷執行。換句話說，子程序A01_1在呼叫其他較低階(低子程序A01_1本身所在的第一階)子程序後，可根據這些子程序回傳之值，判斷周邊裝置32之是否發生運作錯誤，並在錯誤記錄碼 _bErrorCode中記錄運作錯誤發生之情形(像是在執行哪一個低階子程序時發生運作錯誤)，就像在程式區段50A、50B、50C中一樣。相對地，若周邊裝置32在執行子程序A01_1所呼叫的各個低階子程序時均沒有發生錯誤，子程序A01_1就會一直被順利執行至程式區段50D，而在錯誤記錄碼 _bErrorCode[1]中記錄常數READY之值，代表子程序A01_1所呼叫的所有低階子程序均順利完成，而子程序A01_1本身也會回傳常數READY之值，代表子程序A01_1本身也已經順利完成，並結束子程序A01_1之執行。請注意在子程序A01_1中，在記錄錯誤記錄碼 _bErrorCode時所用的程式指令

「 _bErrorCode[_bLevel--]=… 」不僅在錯誤記錄碼 _bErrorCode[1]中記錄了對應常數值，也將變數 _bLevel之值減1，使其回復到子程序A01_1被執行前之值。如圖五A所示，在以程式指令

「 _bErrorCode[_bLevel--]=… 」設定錯誤記錄碼之值後，接下來就會結束子程序A01_1之執行，以程式指令

「 Return(…)」回傳對應常數值，故將變數 _bLevel之值減 1，就可將此變數之值回復到子程序 A01_1本身被呼叫前之值，與子程序 A01_1一開始時之程式指令「 ++_bLevel」對應。

如圖五 B所示，在屬於第二階之子程序 B01_2中，也呼叫了較低階（第三階至第五階）的其他子程序，像是子程序 C01_3。不過，在子程序 B01_2開始時，會先將全域變數 _bLevel加 1，使變數 _bLevel成為 2（因為子程序 B01_2是被子程序 A01_1所呼叫，而變數 _bLevel在子程序 A 1_1執行期間已被設為 1），代表韌體程式碼 46執行至一第二階子程序。就如圖五 B中的程式區段 50E所示，若是周邊裝置 32在執行子程序 B01_2所呼叫的子程序 C01_3時發生運作錯誤，子程序 C01_3就不會傳回常數 READY之值，而子程序 B01_2也就會在錯誤記錄碼 _bErrorCode[2]中記錄常數 C01_Err之值，代表周邊裝置 32在進行子程序 C01_3對應之運作時發生運作錯誤。接下來子程序 B01_2本身也就結束執行，回傳常數值 (!READY)，代表子程序 B01_2在執行期間發生運作錯誤。相對地，若子程序 B01_2順利完成，一直進行至程式區段 50F，就會在錯誤記錄碼 _bErrorCode[2]中記錄常數 READY之值，並在恢復變數 _bLevel之值後，結束子程序 B01_2。請注意，因為子程序 C01_3為一更低階的子程序，其運作的結果會被記錄於陣列變數之錯誤記錄碼 _bErrorCode的次一元素（也

就是 _bErrorCode[2]) 。換句話說，本發明可利用錯誤記錄碼 _bErrorCode的不同元素（或可視為一表列資料型態下的不同欄位）來記錄在不同階子程序所發生的運作錯誤。像是在圖五 A、五 B中，當周邊裝置 32在執行子程序 A01_1所呼叫子程序 B01_2時，若在進行子程序 B01_2呼叫之子程序 C01_3時發生運作錯誤，不但子程序 B01_2會設定錯誤記錄碼 _bErrorCode[2]為常數 C01_Err，子程序 A01_1也會設定錯誤記錄碼 _bErrorCode[1]為常數 B01_Err。依此類推，錯誤記錄碼就能以陣列變數之不同元素，清楚記錄各階子程序運作錯誤發生的情形。

如圖五 C所示，子程序 C01_3中呼叫了較低階的子程序 D01_4。圖五 D中之子程序 D01_4則呼叫了最低階的子程序 E01_5、 E02_5。當然，在這些子程序中，也可以有相對應的程式區段來設定變數 _bLevel之值，並將各子程序運作的情形記錄於錯誤記錄碼 _bErrorCode。熟知技術者應已能由圖五 A、五 B之討論得知相關細節實施的方式；在不妨礙本發明技術揭露的情形下，不再贅述。除了像圖五 A至五 D所示，可在錯誤記錄碼 _bErrorCode[1]、_bErrorCode[2]等元素中分別記錄不同階層子程序運作錯誤發生的情形之外，本發明也可利用錯誤記錄碼 _bErrorCode[0]來記錄最高階（也就是第一階）子程序運作錯誤發生的情形。對照圖五 A所示，在子程序 A01_1執行結束後，就可將一代表子程序 A01_1的代碼記錄於錯

誤記錄碼 _bErrorCode[0]。舉例來說，若周邊裝置32在執行子程序A01_1期間，在執行到子程序B01_2中的子程序C01_3時發生運作錯誤，子程序B01_2、子程序A01_1就會依序中止執行，而錯誤記錄碼 _bErrorCode[0]、_bErrorCode[1]、_bErrorCode[2]記錄的就分別為一常數值A01(代表是在進行子程序A01_1進行期間發生錯誤)、常數B01_Err之值、常數C01_Err之值。當然，某些簡單、低階的子程序也可能不必回傳運作的結果，或是不必將是否發生運作錯誤記錄於錯誤記錄碼。

如前所述，本發明的特點之一，就是使用錯誤處理子程序EH來統一處理周邊裝置32在各子程序進行期間所發生的運作錯誤，以進行對應的錯誤回復。請參考圖五E。圖五E列示的程式碼即為圖四中錯誤處理子程式EH(其子程序之名稱為ErrorHandler)的示意例。在最高階的子程序(舉例來說，第一階子程序A01_1)執行完畢後，本發明就會繼續執行錯誤處理子程序EH，對子程序A01_1執行期間所發生的運作錯誤進行對應的錯誤回復。參照圖五A可知，在子程序A01_1結束後，變數 _bLevel之值應該已回復為0。而如圖五E所示，當子程序A01_1結束、錯誤處理子程序EH開始執行之初，錯誤處理子程序EH會先檢查錯誤記錄碼 _bErrorCode[0]之值，判斷是否有運作錯誤需要進行錯誤回復。若有運作錯誤，則進行至程式區段50G，根據之前進行的第一階子程序為何，判

斷應進行之錯誤回復。其中變數 _bFunctionCode即用來代表之前進行之第一階子程序，而此變數之值即記錄於錯誤記錄碼 _bErrorCode[0]。舉例來說，若變數 _bFunctionCode之值為常數 A01，代表之前進行的為子程序 A01_1，應進行至程式區段 50H，以進行對應子程序 A01_1的錯誤回復；若為另一常數 A02，代表之前進行的是子程序 A02_1，應進行至程式區段 50I，進行對應子程序 A02_1的錯誤回復，以此類推。如圖五 E中的示意例，在確定之前進行的子程序為 A01_1後，在程式區段 50H，還會進一步檢查錯誤記錄碼 _bErrorCode[1]中記錄之值；舉例來說，若錯誤記錄碼 _bErrorCode[1]中記錄的是常數 B01_Err之值，就可於程式區段 50J中進一步確認錯誤記錄碼 _bErrorCode[2]中記錄的錯誤情況。就像圖五 E中程式區段 50J所定義的，若錯誤記錄碼 _bErrorCode[2]中記錄的是常數 C01_Err，錯誤處理子程序 EH就會呼叫另一子程序 B07_2，操控周邊裝置 32進行對應的錯誤回復。換句話說，當周邊裝置 32在執行子程序 A01_1中的子程序 B01_2而於進行子程序 C01_3時發生運作錯誤後，應該要進行子程序 B07_2來進行對應的錯誤回復（子程序 B07_2也就是一回復子程序）。而錯誤處理子程序 EH就會在子程式 A01_1中止後，依據錯誤記錄碼 _bErrorCode中記錄的錯誤發生情況來呼叫子程序 B07_2，完成必要的錯誤回復。

歸納上述描述可知，本發明中是以錯誤處理子程序EH統一管理各種運作錯誤的對應回復運作。錯誤處理子程序 EH會根據錯誤記錄碼 _bErrorCode中記錄之運作錯誤發生情形（像是在哪一階層的哪一個子程序發生運作錯誤），呼叫對應該種情形的回復子程序來進行錯誤回復。換句話說，所有運作錯誤對應的錯誤回復運作都已經依照運作錯誤發生之情形分門別類地記錄於錯誤處理子程序中，便於錯誤回復之集中管理。這樣一來，就不必像習知技術一般，在子程序中各自處理錯誤回復，導致錯誤回復之執行複雜、重複，或無法有效完成。請參考圖五 F。圖五 F與圖四相同，顯示的是韌體程式碼 46中各子程序之架構。總結圖五 A至圖五 E之各個子程序，其執行之流程就如圖五 F所示。首先，由介面程式組 IF中的程式依主機 30（見圖三）之控制指令而要進行伺服程式組 SR中的子程序 A01_1，以操控周邊裝置 32進行對應的運作。如箭頭 F1所示，子程序 A01_1開始被執行，再依箭頭 F2的指示，執行至子程序 B01_2或 B02_2（如圖五 A所示）；以下假設是執行子程序 B01_2。而子程序 B01_2呼叫了較低階的子程序 C01_3、子程序 C01_3呼叫了較低階的子程序 D01_4、子程序 D01_4又呼叫了最低階的子程序 E^1_5、E02_5，其執行流程就如箭頭 F3至 F11所示。子程序 B01_2執行完成後，就可繼續執行子程序 A01_1所呼叫的子程序 B03_2、B04_2，如箭頭 F12至 F13所示。在執行完子程序 A01_1後，執行流程就會如箭頭 F14所指示進行

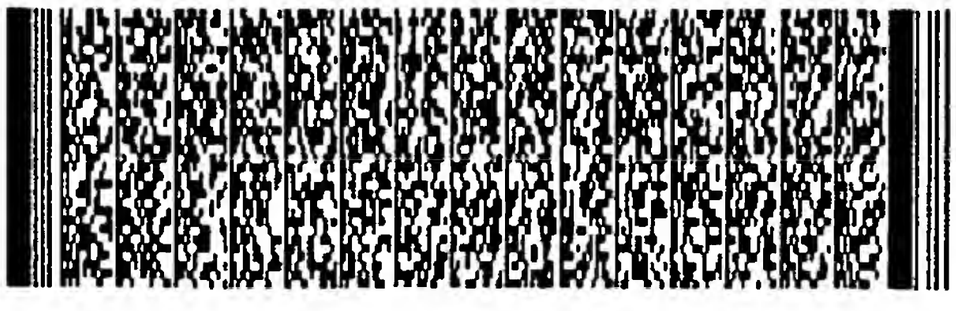
至錯誤處理子程式EH，由錯誤處理子程式EH來針對子程序A01_1執行期間所發生的運作錯誤進行對應的錯誤處理。在錯誤處理子程式EH完成後，執行流程就能如箭頭F17之指示回到介面程式組IF。在錯誤處理子程式EH運作期間，錯誤處理子程式EH也會依照錯誤回復運作的情形重設錯誤記錄碼_bErrorCode。舉例來說，若錯誤回復運作順利，錯誤處理子程式EH就可將原本記錄有錯誤發生情形的錯誤記錄碼_bErrorCode，改記為沒有錯誤發生的情形。或者，有某些錯誤無法由伺服程式組SR中的子程序來完成錯誤回復（像是在周邊裝置32運作期間，使用者突然中斷周邊裝置32之正常運作，周邊裝置32也只能等待使用者的下一個控制指令，才能繼續運作），介面程式組IF也可根據此種情況下的錯誤記錄碼_bErrorCode，將運作錯誤的情形進一步回傳至主機30。

為更進一步說明本發明實際實施的情形，以下將描述本發明之精神實際實施於一可燒錄式光碟機（光碟燒錄器）之情況。換句話說，本發明於圖三中的周邊裝置32為一光碟機，光碟機中的處理器36則執行韌體程式碼46以控制此光碟機之運作。首先請參考圖六。圖六為韌體程式碼46中各子程序架構之示意圖。韌體程式碼46中仍包括了介面程式組IF及伺服程式組SR；以下就以本發明實施於伺服程式組SR之情況做為實施例。就如圖六所示，在本實施例中，伺服程式組SR中的各個子程序被區

五、發明說明 (29)

分為五個階層，在最高階的第一階子程序中包括了子程序 SRVStartUp_1、 SRVCDQSeek_1等等；次階的第二階子程序中有子程序 bReadLeadIn_2等等；第三階子程序則有子程序 PowerOnCalibrate_3等等，第四階中則有子程序 bReadQPosition_4、 bReadATIPPosition_4等等；在最低階的第五階子程序中，則包括有子程序 MediaOKInitSetting_5、 MoveSled_5及 ServoOff_5等等。請注意在各個子程序名稱中最後的數字即用來標示該子程式所屬的階層，像是子程序 ServoOff_5名稱中最後的數字「 5」，就用來表示其為第五階的子程序。在實際是用時，於子程序名稱中加入該子程序所屬之階層，將有助於韌體工程師清楚辨識各子程序的階層，使韌體工程師更容易遵循本發明中子程序依階層序向相互呼叫之原則，也更容易追蹤韌體程式碼執行之流程，或是進行程式除錯。

除了分屬各階層的子程序外，介面程式組 IF可經由一子程序 SRVFunction_0來統一呼叫各第一階子程序，而子程序 ErrorHandler_0就是本發明中的錯誤處理子程序。接下來請繼續參考圖七 A、七 B(並一併參考圖六)。圖七 A列示的即為子程序 SRVFunction_0程式碼之示意例；圖七 B則為子程序 SRVFunction_0執行時之流程圖。圖七 A中也定義了一些常數（像是 ENTRY_LEVEL之值為 0）及巨集（像是巨集 RET）。全域陣列變數



_bErrorCode用來記錄伺服程式組SR中各階層子程序運作錯誤發生的情形。另外，全域變數 _bPlayerStatus也可視為另一個錯誤記錄碼，用來整合記錄各子程序運作錯誤發生的情形。全域變數 _bServoLevel用來記錄韌體程式碼 46執行流程進行至哪一階層的子程序。全域變數 _bErrCnt則用來控制錯誤回復重複進行的次數（將於稍後做進一步說明）。

當韌體程式碼 46之執行流程開始時，介面程式組 IF會根據主機 30之控制指令，設定變數 bFuncName之值，並以七變數之值呼叫子程序 SRVFunction_0；而子程序 SRVFunction_0就會根據變數 bFuncNName之值來呼叫對應的第一階子程序。就如圖七 A、七 B中所示，若變數 bFuncName為一常數 START_UP之值，可代表周邊裝置 32要進行初始化的運作（像是光碟機剛開機時），而子程序 SRVFunction_0就會對應地呼叫第一階子程序 SRVStartUp_1，以操控周邊裝置 32進行初始化的運作，並在錯誤記錄碼 _bErrorCode[0]中記錄常數 START_UP之值。類似地，若變數 bFuncName為一常數 CD_Q_SEEK之值，可代表做為光碟機的周邊裝置 32要進行快速尋軌；而子程序 SRVFunction_0就會對應地呼叫第一階子程序 SRVCDQSeek_1來操控周邊裝置 32進行尋軌，並在錯誤記錄碼 _bErrorCode[0]中記錄常數 CD_Q_SEEK之值，以此類推。在第一階子程序執行完後，子程序 SRVFunction隨即

呼叫錯誤處理子程序ErrorHandler_0，以對第一階子程序進行期間發生之運作錯誤進行錯誤回復。比較特別的是，在子程序SRVFunction_0中，會以程式指令「do…while」之流程控制，來依據子程序ErrorHandler_0進行錯誤處理的情形，控制第一階子程序的重試(retry)。在第一階子程序進行完畢而進行至子程序ErrorHandler_0時，子程序ErrorHandler_0會依據錯誤回復之情況重設錯誤記錄碼_bErrorCode之值，反應錯誤回復進行的情況。子程序ErrorHandler_0執行完畢後，子程序SRVFunction_0就會於程式指令「while」中，根據錯誤記录碼之值（此實施例是依據錯誤記錄碼_bErrorCode[1]之值是否等於一常數EXIT_SRVFUNCTION之值）判斷是否要以程式指令「do… while」重新進行先前呼叫的第一階子程式，進行重試。

上述的重試控制流程也顯示於圖七B的流程中；如圖七B所示，假設介面程式組是以變數bFuncName為常數START_UP之值來呼叫子程序SRVFunction_0，子程序SRVFunction_0就會由圖七中的步驟72A進行至步驟72B、72C；在步驟72C執行子程序ErrorHandler_0時，錯誤記錄碼_bErrorCode也可能會改變（子程序ErrorHandler_0之運作情形將在稍後做進一步說明）。結束步驟72C後，子程序SRVFunction_0就會於程式指令「while」（也就是圖七B中的步驟72D）中依據錯誤記錄碼

_bErrorCode[0]之值而決定接下來的流程。若錯誤記錄碼 _bErrorCode[0]等於常數 EXIT_SRVFUNCTION之值，就能進行至步驟 72E而結束子程序 SRVFunction_0。相對地，若錯誤記錄碼 _bErrorCode[0]不等於常數 EXIT_SRVFUNCTION之值，子程序 SRVFunction_0就會重新執行步驟 72B、72C，也就是重試；在執行步驟 72C時，子程序 ErrorHandler_0還是會依據重試後運作錯誤發生的情形重設錯誤記錄碼 _bErrorCode之值，再重新進行至步驟 72D，判斷是否還需進行另一次重試，以此類推。

請參考圖八。圖八列示的即為第一階子程序 SRVStartUp_1程式碼的示意例。當子程序 SRVStartUP_1被子程序 SRVFunction_0呼叫而開始執行時，會先將變數 _bServoLevel加 1，代表執行流程已進行至一第一階子程序。接下來子程序 SRVStartUp_1會檢查變數 _fgKEjtPressed之狀態（巨集 ChkStatus之定義請參考圖七 A），此變數是用來代表使用者是否按了周邊裝置 32的光碟片「彈出」鈕。若使用者按了此「彈出」鈕，子程序 SRVStartUp_1就會執行巨集 RET，設定錯誤記錄碼 _bErrorCode[1]之值為常數 TRAY_EJECT之值，重設變數 _ServoLevel之值為 0，並結束子程序 SRVStartUp_1。既然使用者按下了「彈出」鈕，子程序 SRVStartUp_1也就可中止執行。相對地，若使用者未按「彈出」鈕，子程序 SRVStartUp_1接下來就會根據變數 _fgPowerOnInit之

值判斷周邊裝置32開機後是否已進行初始化之設定。若否，則進一步呼叫第三階子程序PowerOnCalibrate_3來操控周邊裝置32進行開機後初始化的相關校正，並呼叫第五階子程序MoveSled_5來進行將讀取頭48B(見圖三)移動至某一初始位置。然後子程序SRVStartUp_1又會呼叫第五階子程序CheckMotorStop_5來檢查主軸馬達48A(圖三)是否已經開始轉動，等等。

如圖八中的程式區段52A所示，子程序SRVStartUp_1也會依據一變數_fgATIP之值來判斷光碟片48C的種類(子程序SRVStartUp_1可在之前先呼叫另一較低階的子程序以設定此變數之值)；若周邊裝置32在光碟片48C上讀不出可燒錄式光碟片上獨具之預建軌跡的訊號(也就是ATIP, Absolute Time In Pre-Groove)，代表此光碟片為唯讀光碟片(也就是一般的CD, Compact Disk)。此時子程序SRVStartUp_1就可以呼叫第四階子程序bReadQPosition_4來讀取唯讀光碟片上的Q訊號。一般來說，在光碟片用來記錄資料的軌道上，會區分出不同的資料框架(frame)，用來記錄一定量的資料。而每個資料框架都有其位址；而光碟機即是利用由唯讀光碟片上讀出的Q訊號來定址每個資料框架，以依據各資料框架的位址找出某個特定的資料框架。若周邊裝置32在進行子程序bReadQPosition_4對應之運作時發生運作錯誤(即無法由Q訊號中解析出資料框架定址的資訊，像是光碟片本

身有刮損），就會反映在子程序 bReadQPosition_4回傳之值，而子程序 SRVStartUp_1就會執行至巨集 RET，在錯誤記錄碼 _bErrorCode[1]中記錄常數 bReadQPosition_Err之值，反應子程序 bReadQPosition運作之錯誤，並中止子程序 SRVStartUp_1。另一方面，若由變數 _fgATIP可知光碟片 48C為可燒錄式光碟片（像是 CD-R、CD-RW光碟片），子程序 SRVStartUP_1就會呼叫第四階子程序 bReadATIPPosition_4，以操控周邊裝置 32讀取 ATIP訊號。相對於唯讀光碟片中的Q訊號，在可燒錄式光碟片中，光碟機則是由可燒錄式光碟片上讀出的 ATIP訊號來為可燒錄式光碟片中的資料框架定址。若在子程序 bReadATIPPositoin_4回傳之值反應周邊裝置 32發生了運作錯誤（無法由 ATIP訊號中解析出資料框架定址的資訊），子程序 SRVStartUp_1也會執行巨集 RET，在錯誤記錄碼 _bErrorCode[1]中記錄常數 bReadATIPPosition_Err，並結束子程序 SRVStratUp_1本身之執行。若子程序 bReadATIPPosition_4順利完成，子程序 SRVStartUp_1會繼續呼叫第二階子程序 bReadLeadin_2，操控周邊裝置 32讀取光碟片 48C上的引入區（Lead-In Area)。若周邊裝置 32在進行子程序 b~eadLeadin_2時發生運作錯誤（也就是光碟機找不到光碟片上的引入區），子程序 SRVStartUp_1就會根據子程序 bReadLeadin_2回傳之值，判斷要進行巨集 RET，以設定錯誤記錄碼 _bErrorCode[1]之值為常數

bReadLeadin_Err之值。

換句話說，當周邊裝置32在執行子程序SRVStartUp_1呼叫之低階（也就是低於第一階）子程序時發生運作錯誤，子程序SRVStartUp_1就可依據這些低階子程序回傳之值設定錯誤記錄碼_bErrorCode之值以反應對應的運作錯誤，並中止子程序SRVStartUp_1本身之執行。相對地，如圖八所示，若周邊裝置32順利完成子程序SRVStartUp_1呼叫的各個低階子程序，在執行第五階子程序MediaOkInitSetting_5以設定周邊裝置32後續操控所用的參數後，子程序SRVStartUp_1就會進行至程式區段52B，在錯誤記錄碼_bErrorCode[1]中記錄常數READY之值，代表子程序SRVStartUp_1順利完成備便，而變數_bLevel之值也被重設為0，最後結束子程序SRVStartUp_1之執行。

請參考圖九A至九C。圖九A、圖九B合其來列示的即為錯誤處理子程序ErrorHandler_0程式碼之示意例，圖九C則是子程序ErrorHandler_0進行時之流程圖。如圖九A、九B所示，子程序ErrorHandler_0會先依據錯誤記錄碼_bErrorCode[0]之值判斷是要對哪一個第一階子程序進行錯誤回復之相關操控。由先前於圖七A之子程序SRVFunction_0可知，子程序SRVFunction_0已在呼叫第一階子程序時，於錯誤記錄碼_bErrorCode[0]中記錄了

其所呼叫的第一階子程序（也就是變數 FuncName之值）。等到進行子程序 ErrorHandler_0時，就能依據錯誤記錄碼 _bErrorCode[0]來對各第一階子程序所可能發生的運作錯誤作初步分類。

針對不同的第一階子程序，子程序 ErrorHandler_0會進一步根據錯誤記錄碼 _bErrorCode[1]之值定義對應的錯誤回復。像是在圖九 A、九 B所示的示意例中所顯示的，在子程序 SRVStartUp_1執行完畢後（對應錯誤記錄碼 _bErrorCode[0]為常數 START_UP之值），錯誤記錄碼 _ ErrorCode[1]之值可能為常數 READY、 TRAY_EJECT、 bReadQPosition_Err、 bReadATIPPosition_Err、 bReadLeadin_Err等等之值。對照圖八中的子程序 SRVStartUp_1可知，當周邊裝置在執行子程序 SRVStartUp_1的低階子程序時發生運作錯誤，就會在錯誤記錄碼 _bErrorCode[1]中記錄相對應的常數值。以記錄於錯誤記錄碼 _bErrorCode[1]之值做為索引，子程序 ErrorHandler_0就能找出這些運作錯誤個別所對應的錯誤回復運作。舉例來說，如圖九 A、九 B所示，若錯誤記錄碼 Error_Code[1]中為常數 READY之值，代表周邊裝置3°在執行子程序 SRVStartUp_1期間沒有發生運作錯誤，子程序 ErrorHandler_0就會在錯誤記錄碼 _bErrorCode[0]中記錄常數 EXIT_SRVFUNCTION之值，並在另一個做為錯誤記錄碼的變數 _bPlayerStatus中記錄

常數READY之值，代表周邊裝置32已順利完成子程序SRVStartUp_1之執行；而後子程序ErrorHandler_0就會結束。相對地，若錯誤記錄碼_bErrorCode[1]為常數TRAY_EJECT，代表使用者按下了「彈出」鈕（請對照圖八及相關說明），子程序ErrorHandler除了設定錯誤記錄碼_bErrorCode之值外，也會在變數_bPlayerStatus中記錄常數TRAY_EJECT之值，代表周邊裝置32現在狀況為光碟片匣彈出之狀況；接下來子程序ErrorHandler_0就能結束。

另外，子程序SRVStartUp_1中呼叫的低階子程序可能會呼叫其他更低階的子程序，並在錯誤記錄碼中記錄了更多有關低階子程序運作錯誤之資訊。舉例來說，如圖九B中的程式碼所示，當錯誤記錄碼_bErrorCode[1]中為常數bReadLeadin_Err之值時，子程序ErrorHandler_0還會進一步根據錯誤記錄碼_bErrorCode[2]是常數bSeekATIP_Err或是ReadLeadinInfo_Err等等之值來定義不同的回復運作。像是當錯誤記錄碼_bErrorCode[2]為常數bSeekATIP_Err之值時，子程序ErrorHandler_0還會再進一步地根據錯誤記錄碼_bErrorCode[3]為常數FOCUS_ERROR或是READATIP_ERROR等等之值來判斷周邊裝置32應進行之回復運作。換句話說，若光碟機讀不到光碟片上的引入區（對應於常數bReadLeadin_Err），其可能的原因可能是光碟機無法根據ATIP訊號尋軌（對應於

常數 bSeekATIP_Err) 或是無法正確解析出引入區的資訊（對應於常數 ReadLeadinInfo_Err)。若是光碟機無法根據 ATIP訊號尋軌，其可能的原因還包括了光碟機無法正確將讀取頭雷射據焦於光碟片上（對應於常數 FOCUS_ERROR)，或是無法讀到 ATIP訊號（對應於常數 READATIP_ERROR) 等等。整體來說，針對各種錯誤運作可能發生的情形，其對應之錯誤回復運作都已經詳細定義於本發明中之子程序 ErrorHandler_0，以集中管理錯誤回復的實施。

除了上述在子程序 ErrorHandler_0中依據錯誤記錄碼之內容找出對應之錯誤回復運作外，如前面於圖七 A、七 B討論時所提到的，在子程序 SRVFunction_0執行期間，也可配合錯誤記錄碼 _bErrorCode[0]之值來控制第一階子程式重試之進行。如圖九 A、圖九 B之程式碼所示，子程序 ErrorHandler_0會針對不進行重試的情形，在將錯誤記錄碼 _bErrorCode[0]記錄為常數 EXIT_SRVFUNCTION之值後，結束子程序 ErrorHandler_0。舉例來說，當錯誤記錄碼 _bErrorCode[1]為常數 READY或是 TRAY_EJECT時，前者代表子程序 SRVStartUp_1進行順利，當然不需重試；而後者代表使用者按下「彈出」鈕，周邊裝置 32也應該暫停運作。如圖七 A及圖七 B之流程所示，針對這些不需重試的情形，子程序 SRVFunction_0在執行完子程序

ErrorHandler_0後，就會因為錯誤記錄碼 Error_Code[0]為常數 EXIT_SRVFUNCTION之值，而結束子程式 SRVFunction_0之執行。而介面程式組 IF(請見圖六)就可依據錯誤記錄碼 _bErrorCode或是變數 _bPlayerStatus之值，將周邊裝置 32運作之情形回報至主機 30。

另一方面，針對要進行重試的錯誤回復運作，子程序 ErrorHandler_0則不會將錯誤記錄碼 ErrorCode[0]記錄為常數 EXIT_SRVFUNCTION之值。舉例來說，如圖九 A所示，當周邊裝置 32在子程序 SRVFunction_0的操控下結束第一階子程序 SRVStartUp_1之執行而進行至子程序 ErrorHandler_0時，若錯誤記錄碼 ErrorCode[1]為常數 bReadATIPPosition_Err之值，子程序 ErrorHandler_0就會先操控周邊裝置 32進行第五階子程序 ServoOff_5，讓伺服模組 40C(見圖三)先停止運作，並重設變數 _bPlayerStatus，再於一變數 _bErrCnt中加 1(參照圖七A可知，變數 _bErrCnt之初始值應為 0)。而如圖七 A、七 B所示，在執行完子程序 ErrorHandler_0後，子程序 SRVFunction_0就會在程式指令「while」中因為錯誤記錄碼 _bErrorCode[0]並不等於常數 EXIT_SRVFUNCTION而操控周邊裝置 32再度進行子程序 SRVStartUP_1(也就是重試)，並在子程序 SRVStartUp_1執行結束後第二度進行至子程序 ErrorHandler_0。若在重新進行子程序 SRVStartUp_1時，並沒有發生運作上的錯誤(或發生的

運作錯誤不需進行重試，像是使用者按下「彈出」鈕），在第二度進行至子程序 ErrorHandler時，子程序 ErrorHandler自然就會於錯誤記錄碼 _bErrorCode[0]中記錄常數 EXIT_SRVFUNCTION之值，接下來子程序 SRVFunction就會跳出程式指令「do-while」，結束子程序 SRVFunction。

相對地，若周邊裝置 32在重試期間而第二度進行子程序 SRVStartUp_1時，又發生需要重試的運作錯誤（像是再度發生對應於常數 bReadATIPPosition_Err之運作錯誤），而在後續再度進行子程序 ErrorHandler_0時，子程序 ErrorHandler_0就不會將錯誤記錄碼 _bErrorCode[0]之值設為常數 EXIT_SRVFUNCTION之值；在進行對應的錯誤回復時，也會在變數 _bErrCnt中再度累加 1。而此變數 _bErrCnt就是用來控制第一階子程序重試進行之次數的。請注意，在圖九 A、圖九 B的程式碼中，針對不需重試的錯誤處理，子程序 ErrorHandler_0在將錯誤記錄碼 _bErrorHandler_0設定為常數 EXIT_SRVFUNCTION後，就會進行至程式指令「return」，結束子程序 ErrorHndler_0本身之執行。相對的，針對需要進行重試的錯誤處理（像是當錯誤記錄碼 _bErrorCode[1]為常數 bReadATIPPosition_Err時），子程序 ErrorHandler在累加變數 ErrCnt之值後，還會進行至圖九 B中的程式區段 54，檢查變數 _bErrorCnt之值是

否已經累加超過一常數 MAX_ERR_CNT；若是，就代表重試進行的次數已經太多，而子程序 ErrorHandler_0就會在錯誤記錄碼 _bErrorCode[0]中記錄常數 EXIT_SRVFUNCTION之值，以便在韌體程式碼執行流程回到子程序 SRVFunction_0時，強制結束 SRVFunction，並由介面程式組 IF根據錯誤記錄碼等變數，將周邊裝置 32 重試多次但都不能正常運作的情形回報至主機 30。圖九 C 中總結了子程序 ErrorHandler_0的運作情形，其先根據錯誤記錄碼 _bErrorCode[0]來索引對應的錯誤運作（像是步驟 74A）；若錯誤記錄碼 _bErrorCode之值對應的是不需重試的運作錯誤（也就是某些第一預設值，像是當錯誤記錄碼 _bErrorCode[1]為常數 READY或是 TRAY_EJECT之值時），就可由步驟 74B進行至步驟 74C，進行對應的錯誤回復，並在設定錯誤記錄碼 _bErrorCode[0]為常數 EXIT_SRVFUNCTION後，結束子程序 ErrorHandler_0。若錯誤記錄碼 _bErrorCode之值對應的是需要重試的運作錯誤（也就是第二預設值，像是當錯誤記錄碼 _bErrorCode[1]符合常數 bReadQPosition_Err或是 bReadATIPPosition_Err之值時），就可由步驟 74D進行至步驟 74E，在根據錯誤記錄碼 _bErrorCode進行對應錯誤可復後，累加變數 _bErrCnt之值，並在步驟 74F（對應於圖九 B中之程式區段 54）中檢查變數 _bErrCnt之值是否大於預設之常數 MAX_ERR_CNT，以進行重試次數之控管。

五、發明說明 (42)

總結本發明揭露之原則，是將韌體程式碼中不同的子程序區分為不同的階層，由低階的子程式來定義較簡單、功能較為單一的運作，高階的子程序則呼叫低階的子程序，以定義出較複雜、功能也較完整的運作。而在各子程序間，則以序向的呼叫原則來維持各子程式間相互呼叫的秩序，也就是不由低階子程序呼叫高階子程序，同階層子程序也不互相呼叫（除了最低階的子程序）。另外，本發明還揭露了集中處理錯誤回復運作之原則，利用錯誤記錄碼記錄周邊裝置在執行各階子程序時運作錯誤發生之情形，並統一由一錯誤處理子程式根據錯誤記錄碼來操控周邊裝置進行對應的錯誤回復運作。在習知技術中，因為缺乏對韌體程式碼中各子程式相互呼叫（也就是在某一子程序尚未結束前，又要先執行另一子程式）的管理，會在各子程式間形成複雜的連串呼叫（也就是在某一子程序尚未結束前，又要先執行另一子程式碼的可讀性較低，實際執行時又會耗費大量的處理器資源。而在針對運作錯誤進行錯誤回復時，習知技術也會因為執行流程的複雜、缺乏管理，而造成錯誤回復不必要的重複、不正確的錯誤回復。相較之下，在實施本發明執行流程的單純，使連串呼叫的次數受到向呼叫能維持執行流程的單純，使連串呼叫的次數受到有效的控制，不僅能使韌體程式碼具有較高的可讀性，易於管理、追蹤、除錯，在周邊裝置實際執行韌體程式碼時，也能減少其所需的處理器資源。另外，本發明利

用錯誤處理子程序以根據各階子程序所記錄的錯誤記錄碼來整合不同運作錯誤所對應的錯誤回復運作，能將所有的錯誤回復運作集中管理，避免錯誤回復不必要的重複，也能確保錯誤回復的正確性。因為錯誤記錄碼中已記錄有各階子程序運作的情形，配合上本發明階層序向呼叫的原則，就相當於將韌體程式碼執行之流程及運作情形記錄於其中，不論是在韌體程式碼開發階段的除錯過程裡，或是在實際對周邊裝置進行故障排除時，韌體工程師都能經由錯誤記錄碼中獲知執行流程的相關訊息，方便除錯排障的進行。在上述所討論的實施例中，雖設韌體程式碼是用於與主機搭配的周邊裝置中，但本發明也可適用於單獨運作的電子裝置中，像是手機、數位相機等等，以有效管理這些裝置中韌體程式碼的結構。在韌體程式碼中，本發明之原則可分別實施於介面程式組及伺服程式組的各個子程序間；換句話說，介面程式組中的各個子程序也可區分為不同的階層，並設立屬於介面程式組的錯誤處理子程序。

以上所述僅為本發明之較佳實施例，凡依本發明申請專利範圍所做之均等變化與修飾，皆應屬本發明專利之涵蓋範圍。

圖式之簡單說明；

圖一為一典型周邊裝置與主機配置之功能方塊圖。

圖二A、二B為圖一韌體程式碼習知結構之示意圖。

圖三為本發明中周邊裝置與主機配置之功能方塊圖。

圖四為圖三中韌體程式碼結構之示意圖。

圖五A至五D列示了圖四中相關子程序程式碼之示意例。

圖五E列示了圖四中錯誤處理子程式程式碼之示意例。

圖五F為圖四中韌體程式碼執行流程之示意圖。

圖六為本發明另一實施例中韌體程式碼結構之示意圖。

圖七A列示了圖六中一主要子程序程式碼之示意例。

圖七B為圖七A中子程序執行流程之流程圖。

圖八列示了圖六中一第一階子程序程式碼之示意例。

圖九A、九B列示了圖六中錯誤處理子程式程式碼之示意例。

圖九C為圖九A、九B中錯誤處理子程式進行流程之流程圖。

圖式之符號說明：

圖式簡單說明

10、 30	主機	12、 32	周邊裝置
14A、 34A	中央處理器	14B、 34B	北橋電路
14C、 34C	南橋電路	14D、 34D	記憶體
14E、 34E	顯示卡	14F、 34F	顯示器
16、 36	處理器	18、 38	硬體電路
20A、 40A	編解碼器	20B、 40B	訊號處理器
20C、 40C	伺服模組	22、 42	緩衝記憶體
24、 45	儲存記憶體	26、 46	韌體程式碼
28A、 48A	主軸馬達	28B、 48B	讀取頭
28C、 48C	光碟片	28D、 48D	滑軌

A1-A21、 F1-F17 箭頭

EH 錯誤處理子程式

IF0、 IF 介面程式組

SR0、 SR 伺服程式組

50A-50J、 52A-52B 程式區段

R3s、 R5s、 72A-72D、 74A-74F 步驟

R1、 R2A-R2B、 R3A-R3B、 R4A-R4B、 R5A-R5C、 R6A-R6B、 R7A-R7B、 R8A-R8B、 R9、 A01_1-A02_1、 B01_2-B07_2、 C01_3-C03_3、 D01_4-D02_4、 E^1_5-E03_5 子程序

六、申請專利範圍

1. 一種以一處理器操控一硬體電路之方法，該處理器係執行一程式碼以操控該硬體電路，其中該程式碼中包含有：

複數個後階子程序，當該處理器執行不同的後階子程序後，會操控該硬體電路進行不同的對應運作，而每一後階子程序會將該硬體電路進行對應運作之結果記錄於一錯誤記錄碼；

其中不同的運作結果對應於不同的回復運作；

複數個前階子程序，各前階子程序用來呼叫至少一後階子程序，使該處理器在執行一前階子程式時，會根據該前階子程式中呼叫的後階子程式操控該硬體電路進行各後階子程式對應的運作；

複數個回復子程序，各回復子程序對應於一回復運作，使得當該處理器執行不同的回復子程序後，會操控該硬體電路進行不同的對應回復運作；以及

一錯誤處理子程序，用來根據該錯誤記錄碼之內容呼叫該等回復子程序；

而該方法包含有：

於該處理器執行前階子程序後，執行該錯誤處理子程序，以使該處理器得以根據前階子程序中各後階子程序對應運作之結果來操控該硬體電路進行對應之回復運作。

2. 如申請專利範圍第 1項之方法，其中當該處理器執行

前階子程序後再執行該錯誤處理子程序時，係使該處理器不會在該前階子程序結束前執行各後階子程序對應之回復運作。

3. 如申請專利範圍第1項之方法，其中各前階子程序不會互相呼叫，使該處理器在執行完一前階子程序之前，不會執行另一前階子程序。

4. 如申請專利範圍第1項之方法，其中該硬體電路為一光碟機的伺服模組，該伺服模組包含有：
一主軸馬達，用來帶動一光碟片轉動；以及
一雷射讀取頭，用來產生一雷射入射至該光碟片。

5. 如申請專利範圍第1項之方法，其中該硬體電路為一光碟機的介面模組。

6. 如申請專利範圍第1項之方法，其中該錯誤記錄碼為該程式碼中的全域(global)變數；而各後階子程序係將對應運作之結果記錄於同一錯誤記錄碼。

7. 如申請專利範圍第1項之方法，其中該程式碼另包含有複數個次階子程序；當該處理器執行不同的次階子程序時，會操控該硬體電路進行不同的運作；每一次階子程序亦會將該硬體電路進行對應運作之結果記錄於一第

二錯誤記錄碼；而各後階子程序係用來呼叫至少一次階子程序，使得當該處理器執行一後階子程序時，會依序執行該後階子程序中的次階子程序以操控該硬體電路進行該後階子程序對應之運作。

8. 如申請專利範圍第7項之方法，其中每一後階子程序中的次階子程序係將對應運作之結果記錄於同一第二錯誤碼。

9. 如申請專利範圍第7項之方法，其中該第二錯誤碼為該錯誤碼中的一個欄位。

10. 如申請專利範圍第7項之方法，其中各次階子程序係將對應運作之結果記錄於同一第二錯誤碼。

11. 如申請專利範圍第7項之方法，其中該第二錯誤碼為該錯誤碼的一個欄位。

12. 如申請專利範圍第1項之方法，其中各後階子程序不會互相呼叫，使得當該處理器在執行完一後階子程序前，不會執行另一後階子程序。

13. 如申請專利範圍第1項之方法，其中各後階子程序不會呼叫該等前階子程序。

六、申請專利範圍

14. 一種電子裝置，其包含有：

一硬體電路，用來實現該電子裝置的功能；

一處理器，用來執行一程式碼以操控該硬體電路；

一儲存記憶體，用來儲存該程式碼；其中該程式碼中包含有：

複數個後階子程序，當該處理器執行不同的後階子程序後，會操控該硬體電路進行不同的對應運作，而每一後階子程序會將該硬體電路進行對應運作之結果記錄於一錯誤記錄碼；

其中不同的運作結果對應於不同的回復運作；

複數個前階子程序，各前階子程序用來呼叫至少一後階子程序，使該處理器在執行一前階子程式時，會根據該前階子程式中呼叫的後階子程式操控該硬體電路進行各後階子程式對應的運作；

複數個回復子程序，各回復子程序對應於一回復運作，使得當該處理器執行不同的回復子程序後，會操控該硬體電路進行不同的對應回復運作；以及

一錯誤處理子程序，用來根據該錯誤記錄碼之內容呼叫該等回復子程序；

其中當該處理器執行前階子程序後，會執行該錯誤處理子程序，以使該處理器得以根據前階子程序中各後階子程序對應運作之結果來操控該硬體電路進行對應之回復運作。

六、申請專利範圍

15. 如申請專利範圍第14項之電子裝置，其中當該處理器執行前階子程序後再執行該錯誤處理子程序時，係使該處理器不會在該前階子程序結束前執行各後階子程序對應之回復運作。

16. 如申請專利範圍第14項之電子裝置，其中各前階子程序不會互相呼叫，使該處理器在執行完一前階子程序之前，不會執行另一前階子程序。

1. 如申請專利範圍第14項之電子裝置，其係為一光碟機，而該硬體電路包含有一伺服模組，其中該伺服模組包含有：
一主軸馬達，用來帶動一光碟片轉動；以及
一讀取頭，用來產生一雷射入射至該光碟片。

18. 如申請專利範圍第14項之電子裝置，其係為一光碟機，而該硬體電路為該光碟機的介面模組。

19. 如申請專利範圍第14項之電子裝置，其中該錯誤記錄碼為該程式碼中的全域(global)變數；而各後階子程序係將對應運作之結果記錄於同一錯誤記錄碼。

20. 如申請專利範圍第14項之電子裝置，其中該程式碼

另包含有複數個次階子程序；當該處理器執行不同的次階子程序時，會操控該硬體電路進行不同的運作；每一次階子程序亦會將該硬體電路進行對應運作之結果記錄於一第二錯誤記錄碼；而各後階子程序係用來呼叫至少一次階子程序，使得當該處理器執行一後階子程序時，會依序執行該後階子程序中的次階子程序以操控該硬體電路進行該後階子程序對應之運作。

21. 如申請專利範圍第20項之電子裝置，其中每一後階子程序中的次階子程序係將對應運作之結果記錄於同一第二錯誤碼。

22. 如申請專利範圍第20項之電子裝置，其中該第二錯誤碼為該錯誤碼中的一個欄位。

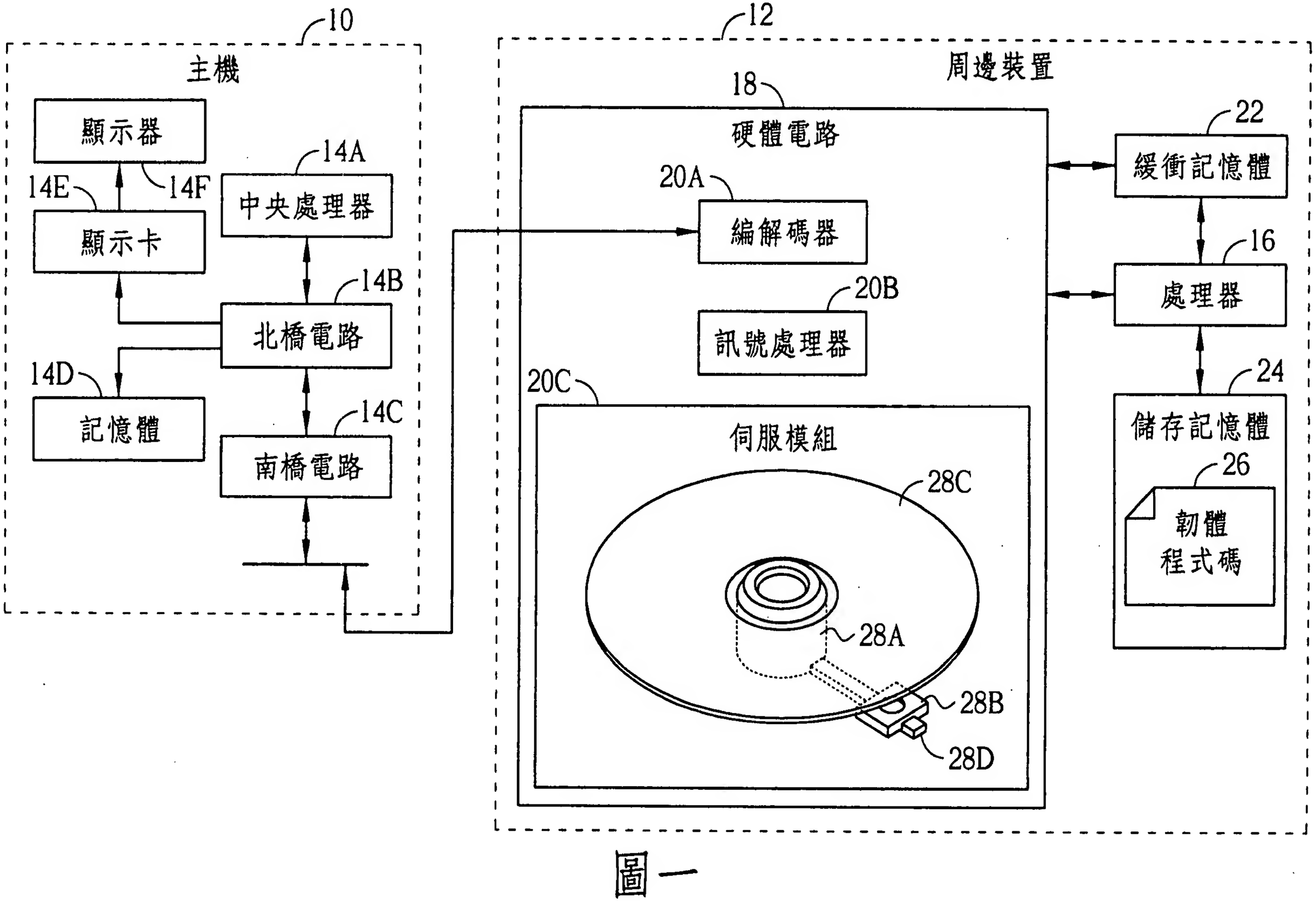
23. 如申請專利範圍第20項之電子裝置，其中各次階子程序係將對應運作之結果記錄於同一第二錯誤碼。

24. 如申請專利範圍第20項之電子裝置，其中該第二錯誤碼為該錯誤碼的一個欄位。

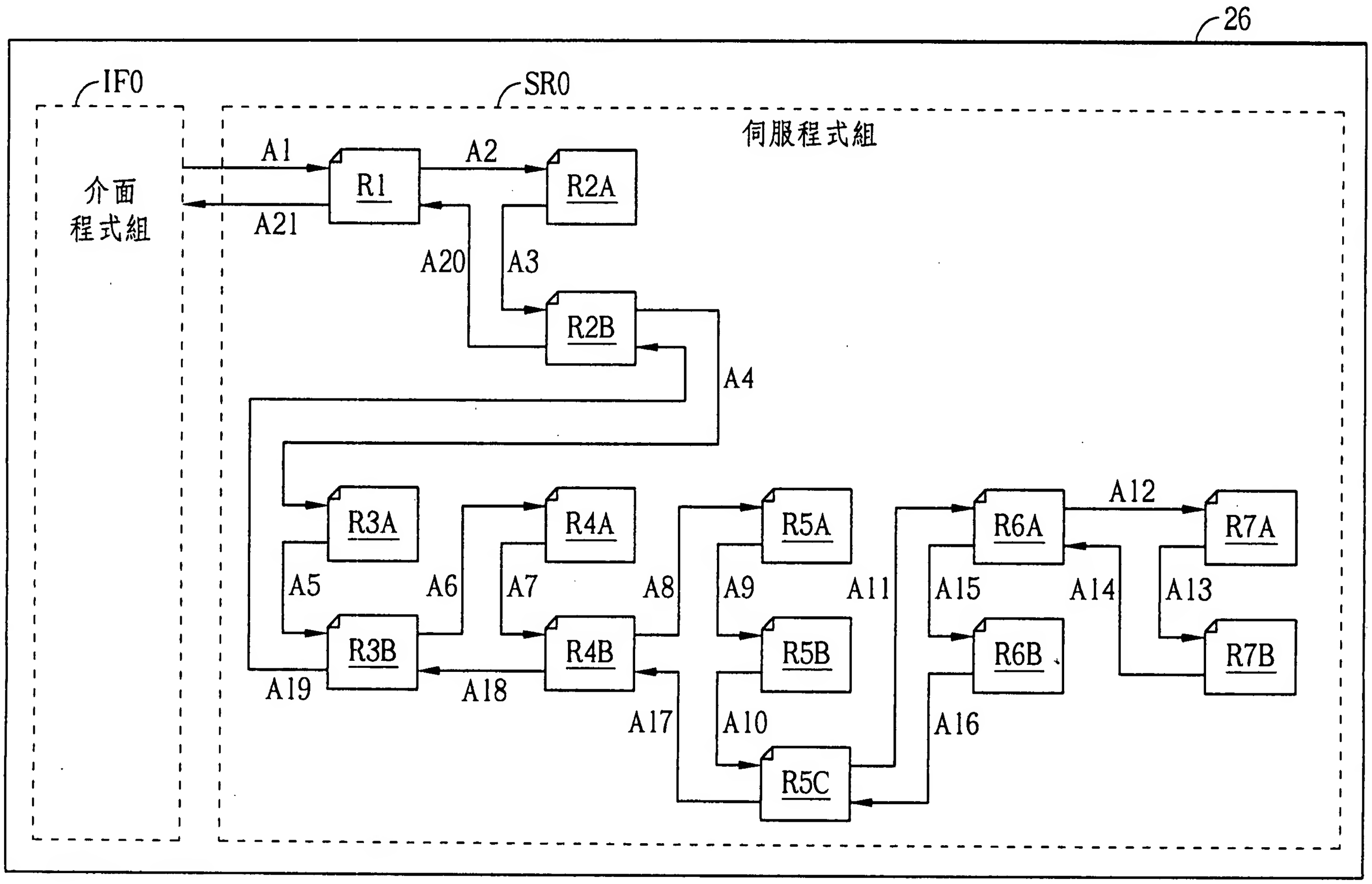
25. 如申請專利範圍第14項之電子裝置，其中各後階子程序不會互相呼叫，使得當該處理器在執行完一後階子程序前，不會執行另一後階子程序。

六、申請專利範圍

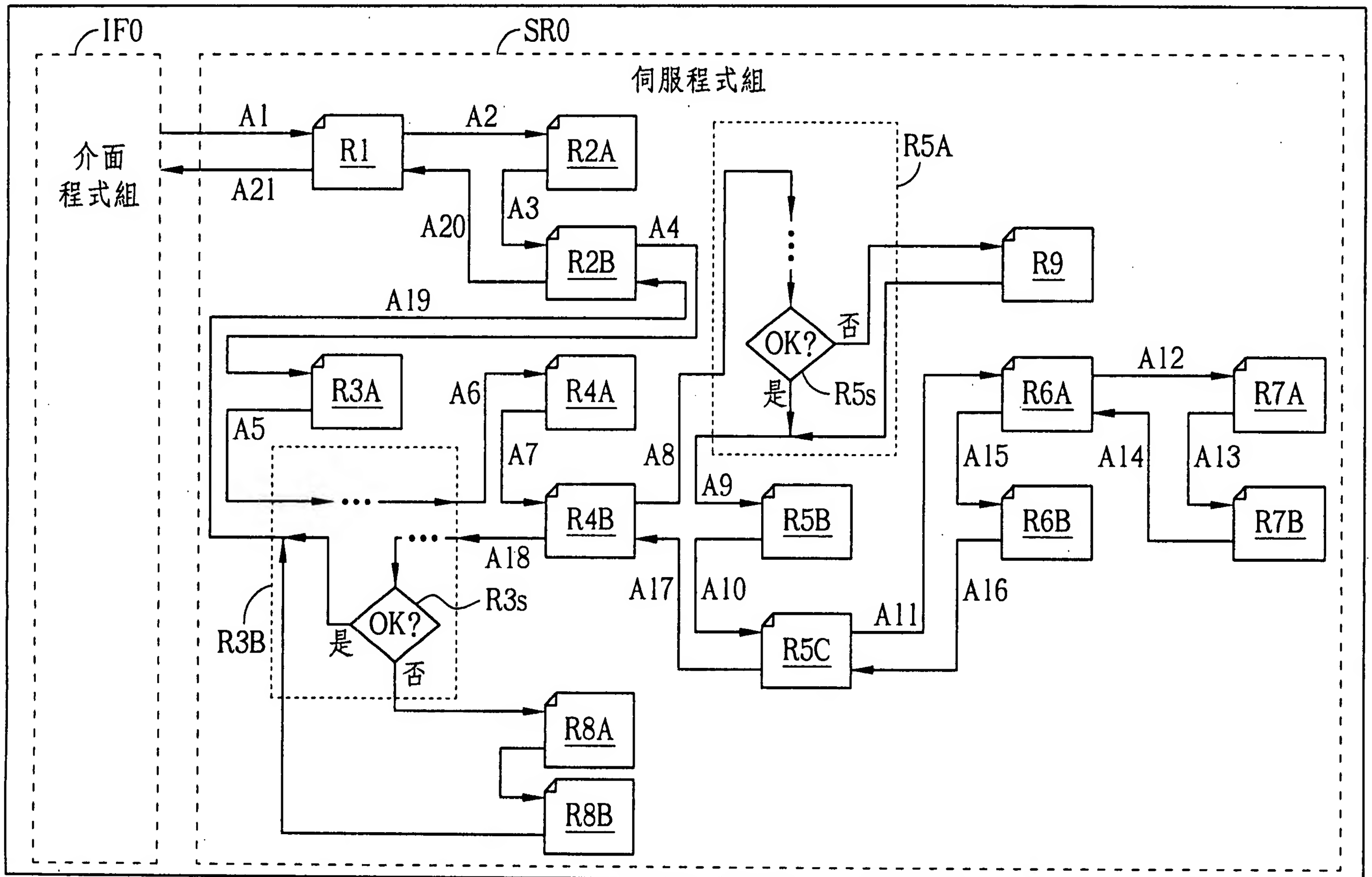
26. 如申請專利範圍第14項之電子裝置，其中各後階子程序不會呼叫該等前階子程序。


10
主機
顯示器
14A
14E
14F
中央處理器
顯示卡
14B
北橋電路
14D
記憶體
14C
南橋電路
12
周邊裝置
18
硬體電路
20A
編解碼器
20B
訊號處理器
20C
伺服模組
28C
28A
28B
28D
22
緩衝記憶體
16
處理器
24
儲存記憶體
26
韌體
程式碼

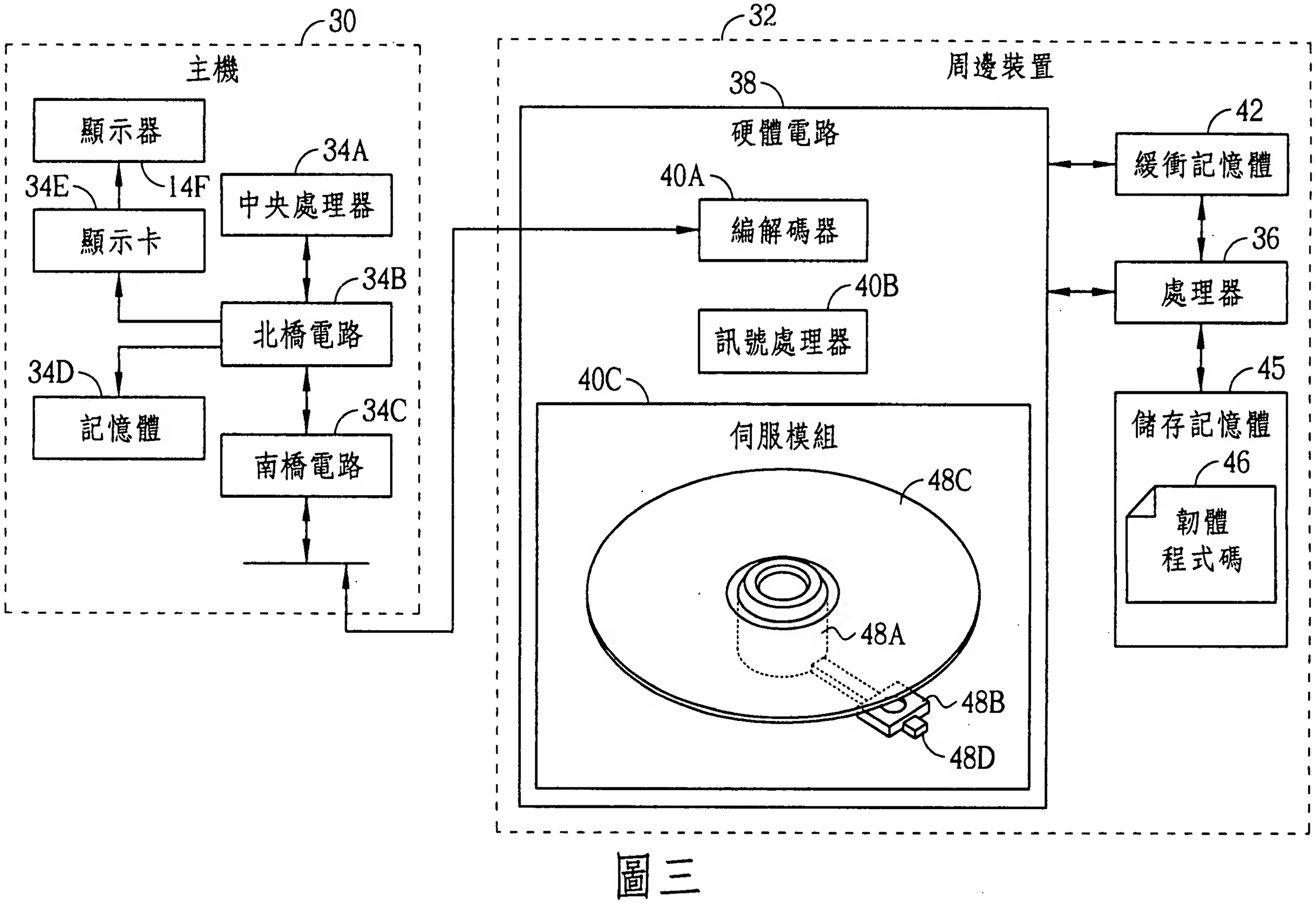
圖一


26
IF0
介面
程式組
SR0
伺服程式組
A1
A2
R1
R2A
A21
A20
A3
R2B
A4
R3A
R4A
R5A
R6A
A12
R7A
A5
A6
A7
A8
A9
A11
A15
A14
A13
R3B
R4B
R5B
R6B
R7B
A19
A18
A17
A10
A16
R5C

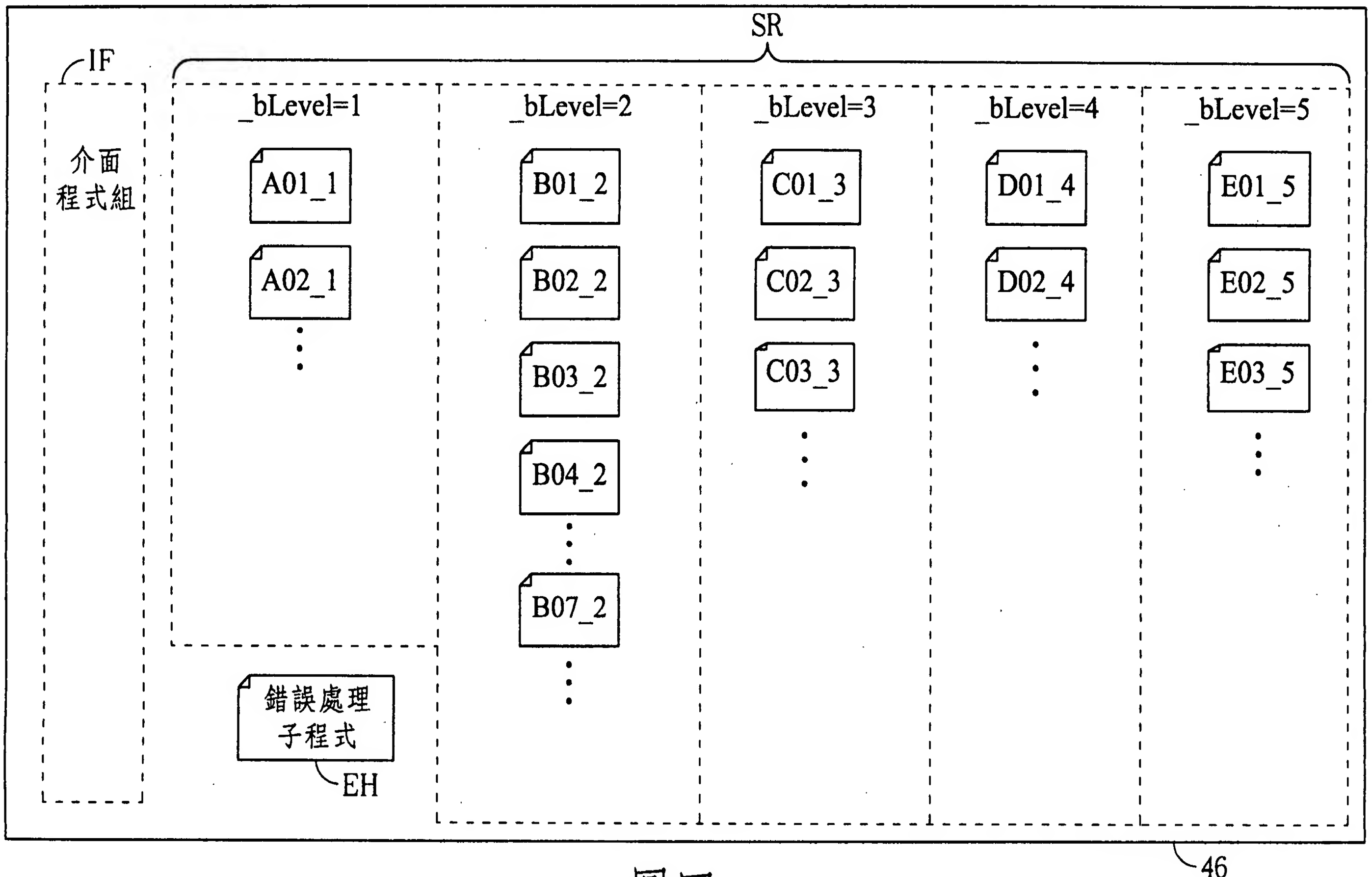
圖二A


IF0
SR0
伺服程式組
介面
程式組
A1
A2
R1
R2A
A21
A3
A20
A4
R2B
A19
R5A
R9
OK?
否
是
R5s
R3A
A6
R4A
A12
R6A
R7A
A5
A7
A8
A15
A14
A13
A9
R5B
R4B
R6B
R7B
A18
A17
A10
A16
A11
R5C
R3s
OK?
R3B
是
否
R8A
R8B
26

圖二B


30
主機
顯示器
14F
34A
34E
中央處理器
顯示卡
34B
北橋電路
34D
34C
記憶體
南橋電路
32
周邊裝置
38
硬體電路
40A
編解碼器
40B
訊號處理器
40C
伺服模組
48C
48A
48B
48D
42
緩衝記憶體
36
處理器
45
儲存記憶體
46
韌體
程式碼

圖三


SR
IF
介面
程式組
_bLevel=1
_bLevel=2
_bLevel=3
_bLevel=4
_bLevel=5
A01_1
A02_1
B01_2
B02_2
B03_2
B04_2
B07_2
C01_3
C02_3
C03_3
D01_4
D02_4
E01_5
E02_5
E03_5
錯誤處理
子程式
EH
46

圖四

```
#define ChkStatus(x)  x
#define SetStatus(x)    x = 1;
#define ClrStatus(x)    x = 0;

BYTE A01_1(...)
{
 ++_bLevel;
 ...
 /* Global status check */
 if (ChkStatus( _fgSelectB01_2))
 {
   if (B01_2(...)!= READY)
   {
    _bErrorCode[ _bLevel--] = B01_Err;
    return(!READY);
   }
}
else
{
   if (B02_2(...) != READY)
   {
    _bErrorCode[ _bLevel--] = B02_Err;
    return(!READY);
   }
...
}
...
if (B03_2(...) != READY)
{
 _bErrorCode[ _bLevel--] = B03_Err;
 return(!READY);
}
...
if (B04_2(...) != READY)
{
 _bErrorCode[ _bLevel--] = B04_Err;
 return(!READY);
}
...
_bErrorCode[ _bLevel--] = READY;
return(READY);
}
```

50A

50B

50C

50D

圖五A

```
BYTE B01_2(...)
{
 ++_bLevel;
 ...
 if (C01_3(...) != READY)
 {
   _bErrorCode[ _bLevel--]=C01_Err;
   ClrStatus( _fgSelectB01_2);
   return(!READY);
 }
 ...
 _bErrorCode[ _bLevel--] = READY;
 return(READY);
}
```

50E

50F

圖五B

```
BYTE C01_3(...)
{
 ...
 D01_4(...);
 ...
   SetStatus( _fgSelectB01_2);
 ...
}
```

圖五C

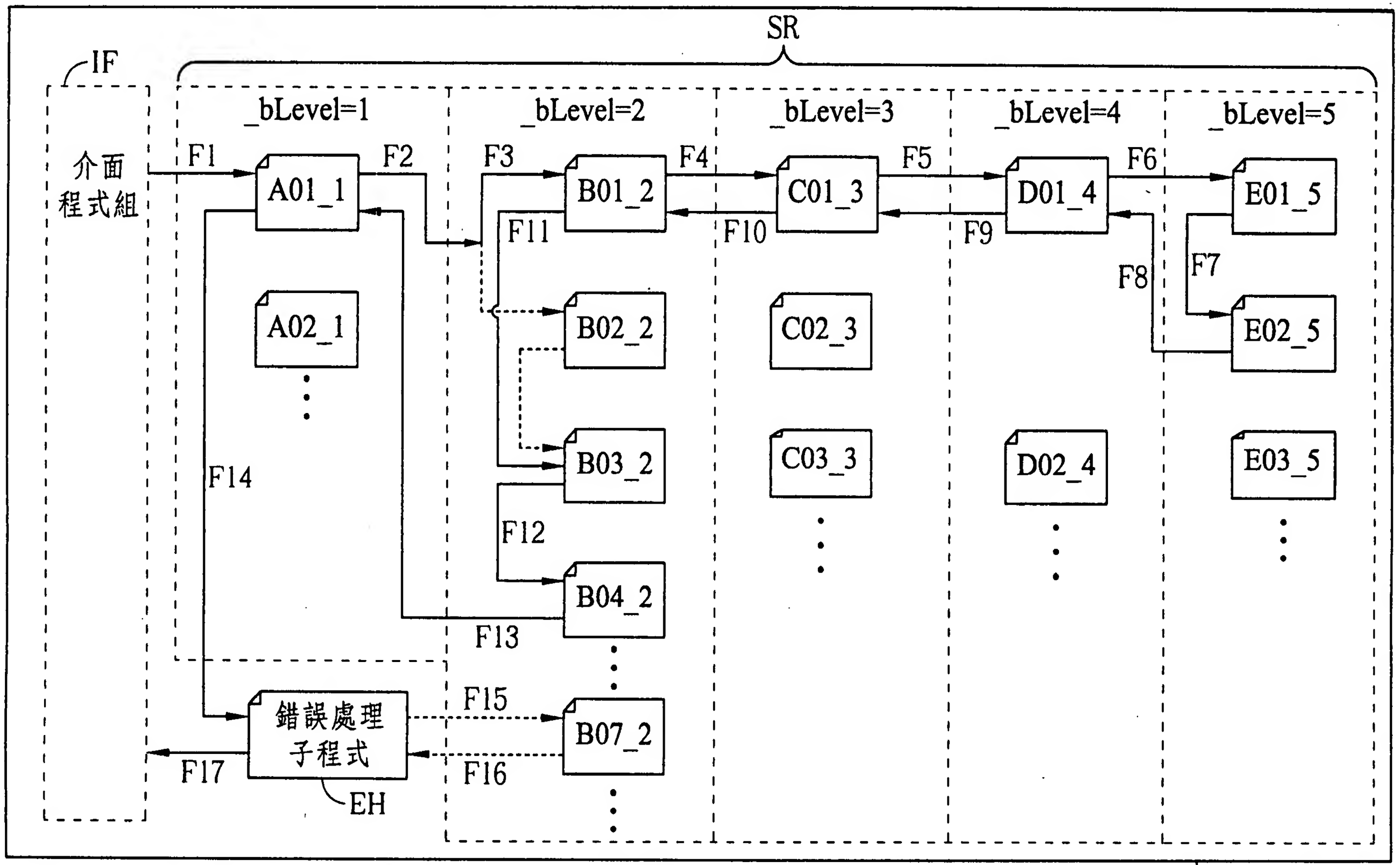
```
BYTE D01_4(...)
{
 ...
 E01_5(...);
 ...
 E02_5(...);
 ...
}
```

圖五D

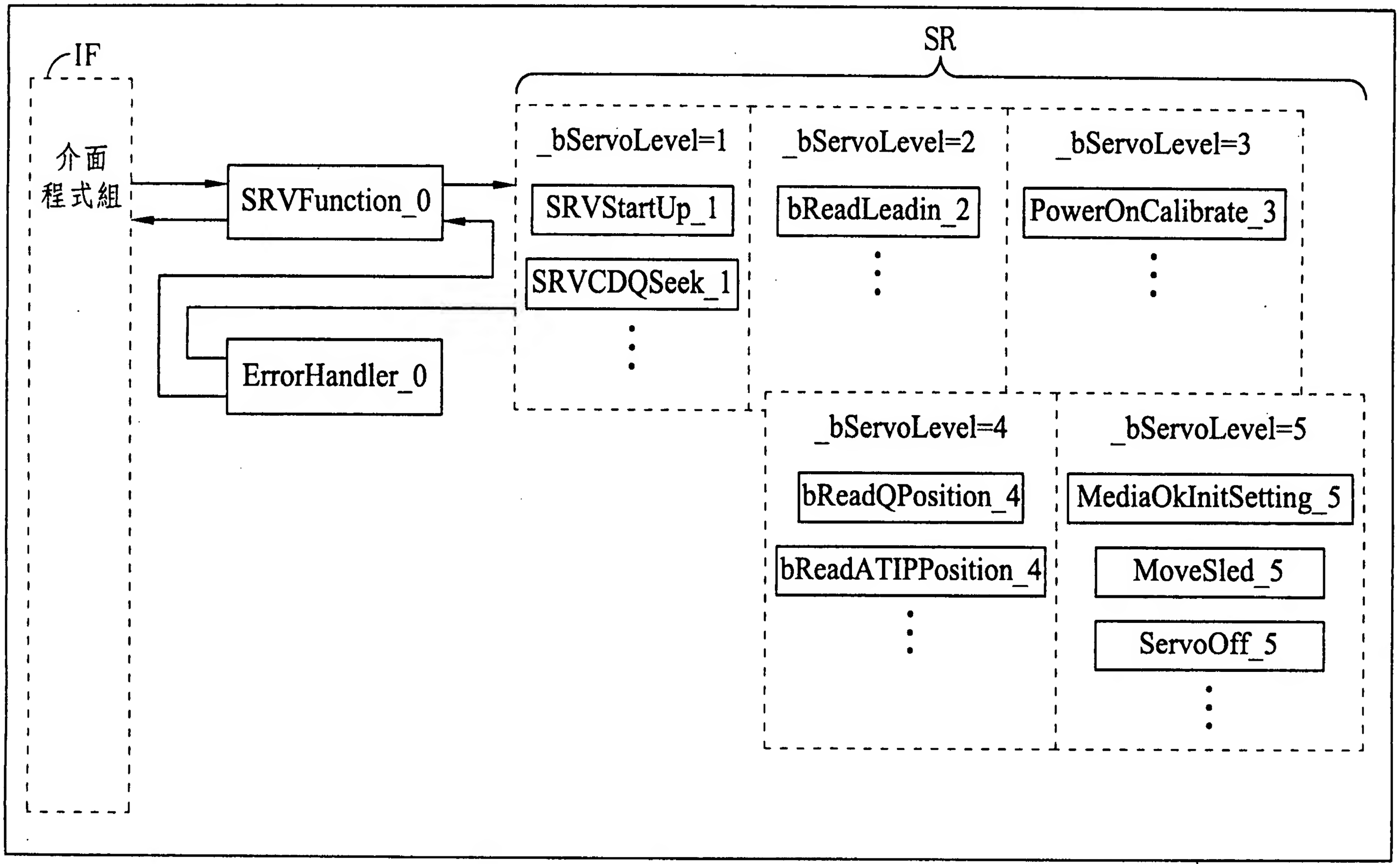
```
void ErrorHandler( )
{
 /* check if error occurs */
 if ( _ErrorCode[ _bLevel] != READY)
 {
  switch ( _bFunctionCode)
  {
   case A01:       // Function A01_1
      switch ( _bErrorCode[ _bLevel+1])
      {
        case READY:
      return;
   case B01_Err:
      switch ( _bErrorCode[ _bLevel+2])
      {
        case READY:
          return;
        case C01_Err:    // error recovery for C01_Err
          B07_2(...);
          return;
        ...
        default:
          return;
      }
    case B02_Err:
      ...
    case B03_Err:
      ...
        default:
          return;
      }
    case A02:       // Function A02_1
    ...
    default:
     return;
    }
   }
  }
}
```

50G 50H 50J 50I

圖五E


SR
IF
介面
程式組
_bLevel=1
_bLevel=2
_bLevel=3
_bLevel=4
_bLevel=5
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
A01_1
A02_1
B01_2
B02_2
B03_2
B04_2
B07_2
C01_3
C02_3
C03_3
D01_4
D02_4
E01_5
E02_5
E03_5
錯誤處理
子程式
EH
46

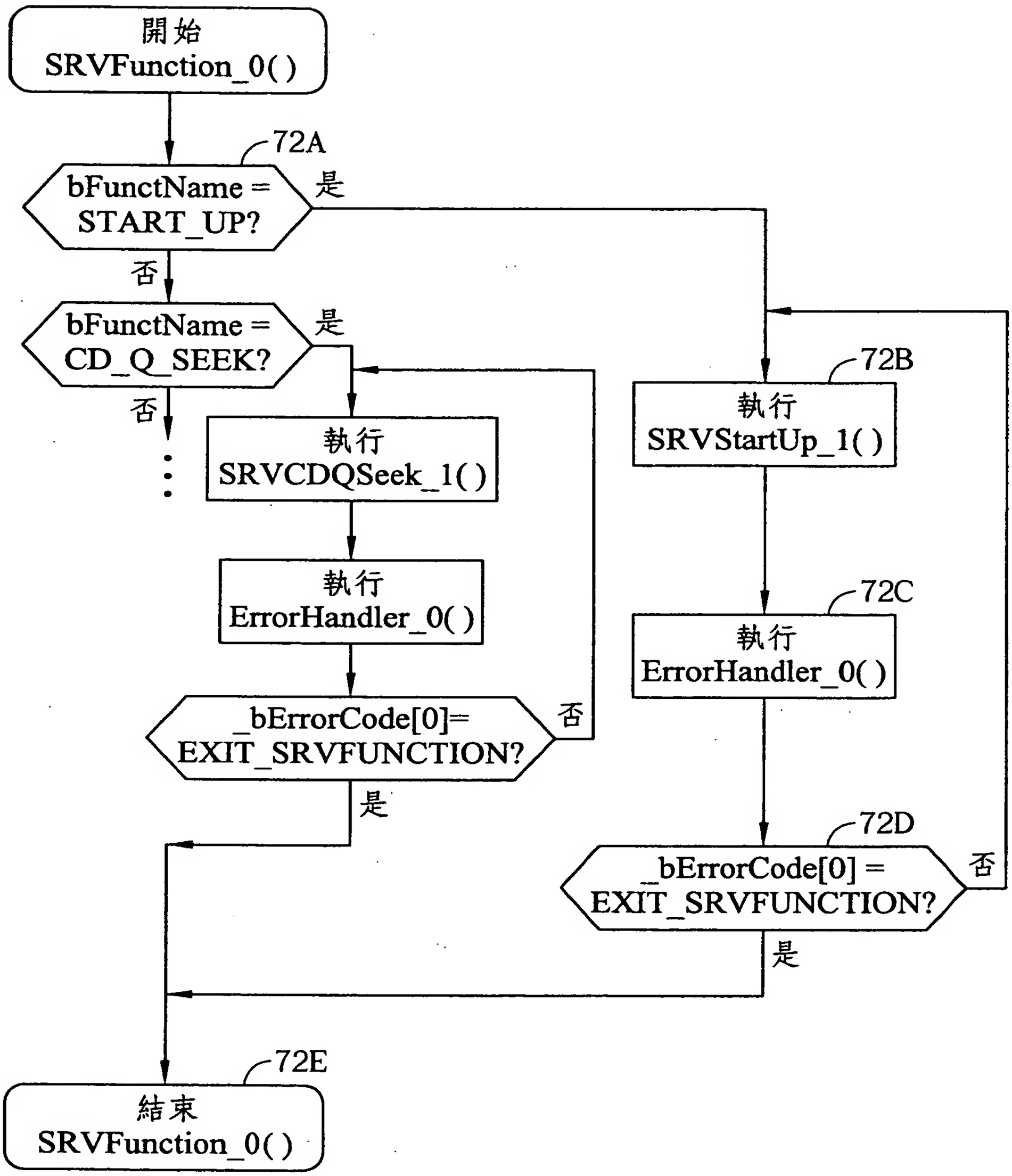
圖五F


IF
介面
程式組
SRVFunction_0
ErrorHandler_0
SR
_bServoLevel=1
SRVStartUp_1
SRVCDQSeek_1
_bServoLevel=2
bReadLeadin_2
_bServoLevel=3
PowerOnCalibrate_3
_bServoLevel=4
bReadQPosition_4
bReadATIPPosition_4
_bServoLevel=5
MediaOkInitSetting_5
MoveSled_5
ServoOff_5
46

圖六

```
#define ENTRY_LEVEL   0
#define MAX_ERR_CNT   3
#define RET(x)    {                                    \
                    _bErrorCode[ _bServoLevel--] = x ; \
                    return;                            \
                  }
#define RET1(x)   {                                    \
                    _bErrorCode[ _bServoLevel--] = x ; \
                    return(x);                         \
                  }
#define ChkStatus(x)  x
#define SetStatus(x)    x = 1;
#define ClrStatus(x)    x = 0;

//-----------------------------------------------
void SRVFunction_0(BYTE bFuncName )
{

 _bServoLevel = ENTRY_LEVEL;
 _bErrCnt = 0;
 _bErrorCode[ _bServoLevel] = bFuncName;

 do
 {
  switch (bFuncName)
  {
   case START_UP:
        SRVStartUp_1( );
        break;

   case CD_Q_SEEK:
        SRVCDQSeek_1( );
        break;

   default:
        break;
  }
  ErrorHandler_0( );

 } while( _bErrorCode[0]!=EXIT_SRVFUNCTION);

}
```

圖七A


開始
SRVFunction_0( )
72A
bFunctName =
START_UP?
是
否
bFunctName =
CD_Q_SEEK?
是
否
執行
SRVCDQSeek_1( )
執行
ErrorHandler_0( )
_bErrorCode[0]=
EXIT_SRVFUNCTION?
否
是
72B
執行
SRVStartUp_1( )
72C
執行
ErrorHandler_0( )
72D
_bErrorCode[0] =
EXIT_SRVFUNCTION?
否
是
72E
結束
SRVFunction_0( )

圖七B

```
void SRVStartUp_1( )
{
   _bServoLevel++;

  if (ChkStatus( _fgKEjtPressed ))   RET( TRAY_EJECT );

  if (!ChkStatus( _fgPowerOnInit ))
  {
   PowerOnCalibrate_3( );
   MoveSled_5(TO_INNER, MOVE_SLED_HOME);
  }

  CheckMotorStop_5( );
  ...
  if(!_fgATIP)
  {
   if( bReadQPosition_4( ) )  RET( bReadQPosition_Err );
  }
52A else
  {
   if( bReadATIPPosition_4( ) )  RET( bReadATIPPosition_Err );
   if( bReadLeadin_2( ) )  RET( bReadLeadin_Err );
  }

   MediaOkInitSetting_5( );

52B RET( READY );
}
```

圖八

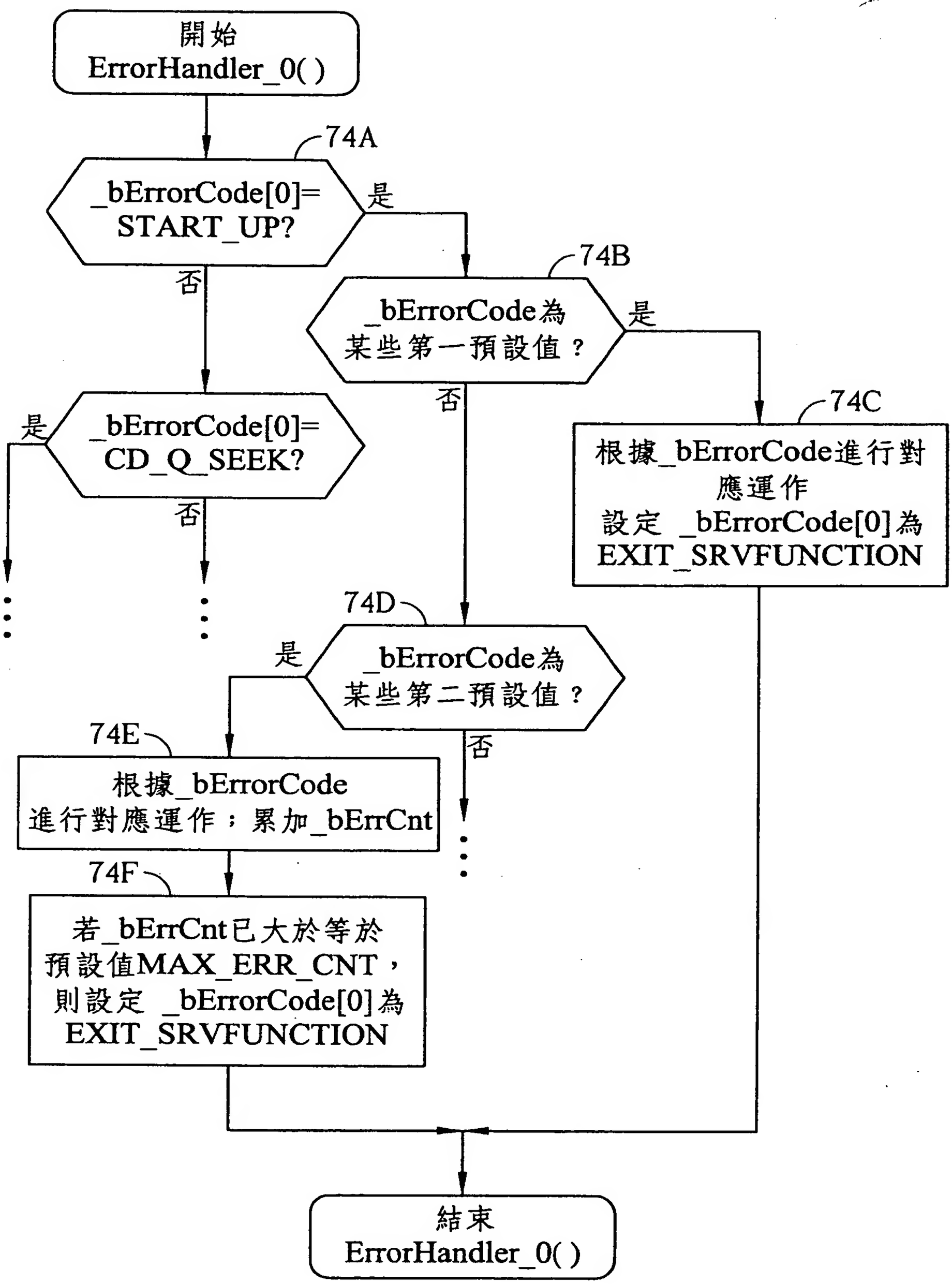
```
void ErrorHandler_0( )
{
   switch ( _bErrorCode[0])
   {
    case START_UP:
       switch ( _bErrorCode[1])
       {
        case READY:
          _bErrorCode[0] = EXIT_SRVFUNCTION;
          _bPlayerStatus = READY;
          return;

       case TRAY_EJECT:
          _bErrorCode[0] = EXIT_SRVFUNCTION;
          _bPlayerStatus = TRAY_EJECT;
          return;

       case bReadQPosition_Err:
          if( _fgDiskIsDVD)
          {
           _bMediaType=CDROM_DISC;
          }
          ServoOff_5( );
          _bPlayerStatus = _bErrorCode[2];
          _bErrCnt++;
          break;

       case bReadATIPPosition_Err:
          ServoOff_5( );
          _bPlayerStatus = _bErrorCode[2];
          _bErrCnt++;
          break;
```

圖九A

```
case bReadLeadin_Err:
  switch( _bErrorCode[2])
  {
      case bSeekATIP_Err:
          _bPlayerStatus =_bErrorCode[3];
          switch( _bErrorCode[3])
          {
            case FOCUS_ERROR:
              ...
            case READATIP_ERROR:
              ...
          }
          break;

        case ReadLeadinInfo_Err:
          ...
    }
    break;

  default:
    break;
 }
case CD_Q_SEEK
  ...
default:
  _bErrorCode[0] = EXIT_SRVFUNCTION;
  _bPlayerStatus = ILLEGAL_COMMAND;
  break;
 }
 if( _bErrCnt >= MAX_ERR_CNT )
 {
  _bErrorCode[0] = EXIT_SRVFUNCTION;
 }
}
```

54

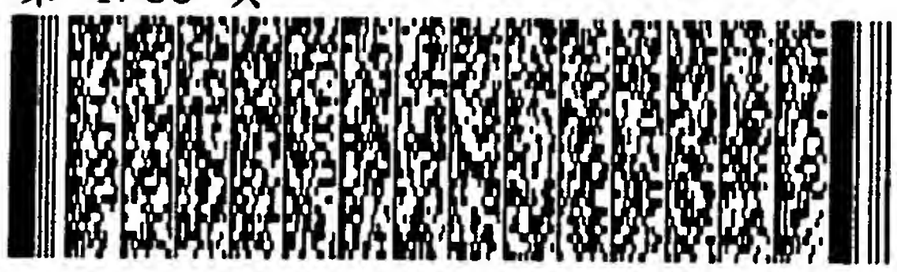
圖九B


開始
ErrorHandler_0( )
74A
_bErrorCode[0]=
START_UP?
是
否
74B
_bErrorCode為
某些第一預設值？
是
否
74C
根據_bErrorCode進行對
應運作
設定 _bErrorCode[0]為
EXIT_SRVFUNCTION
是
_bErrorCode[0]=
CD_Q_SEEK?
否
74D
是
_bErrorCode為
某些第二預設值？
否
74E
根據_bErrorCode
進行對應運作；累加_bErrCnt
74F
若_bErrCnt已大於等於
預設值MAX_ERR_CNT，
則設定 _bErrorCode[0]為
EXIT_SRVFUNCTION
結束
ErrorHandler_0( )

圖九C

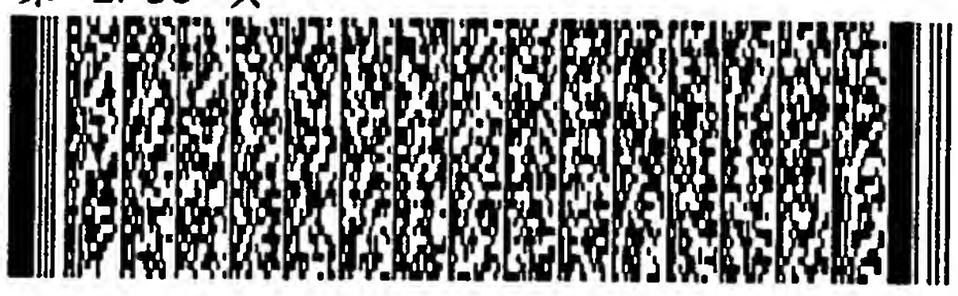
申請案件名稱:統一處理多個子程序執行回應之韌體架構方法及相關裝置

第 1/58 頁

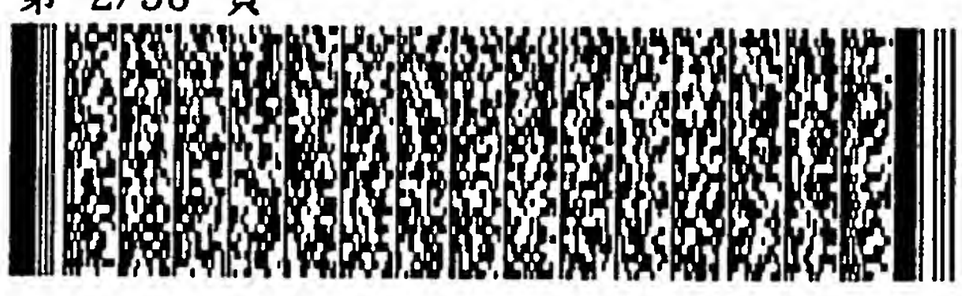


第 1/58 頁

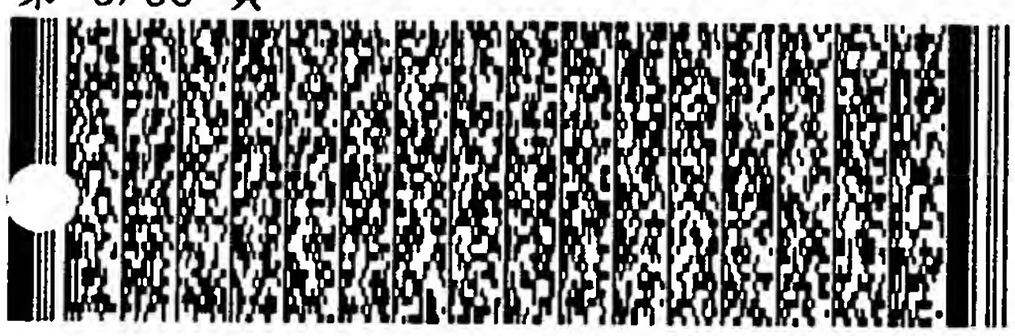
第 2/58 頁



第 2/58 頁



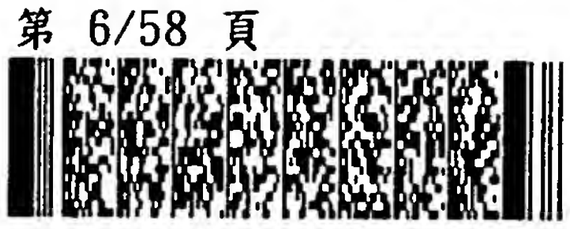
第 3/58 頁



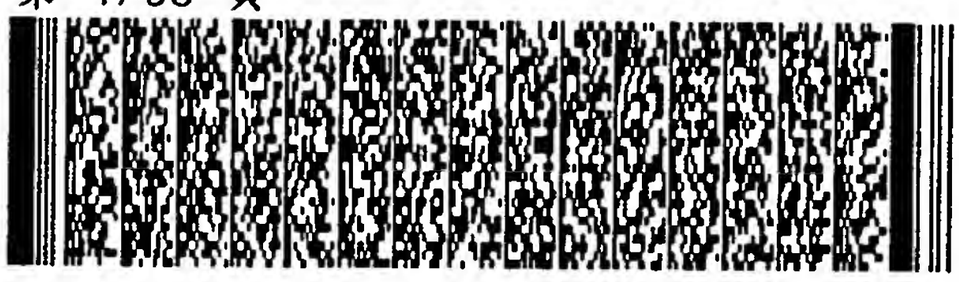
第 4/58 頁

第 5/58 頁

第 6/58 頁

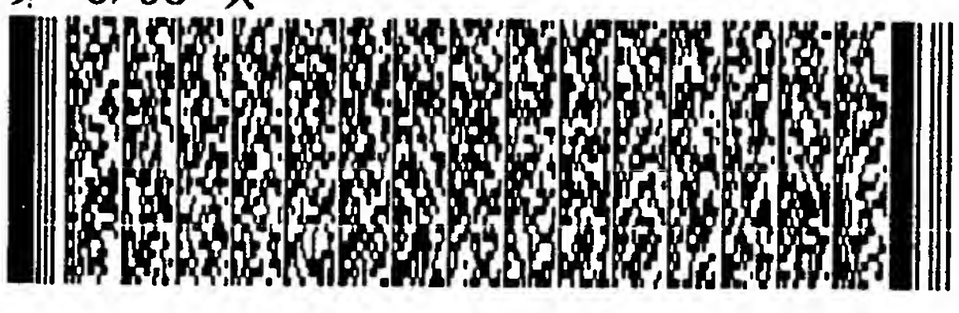


第 7/58 頁



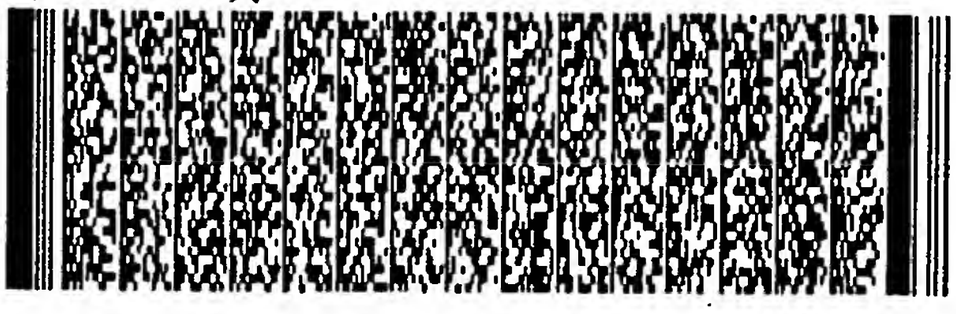
第 7/58 頁

第 8/58 頁

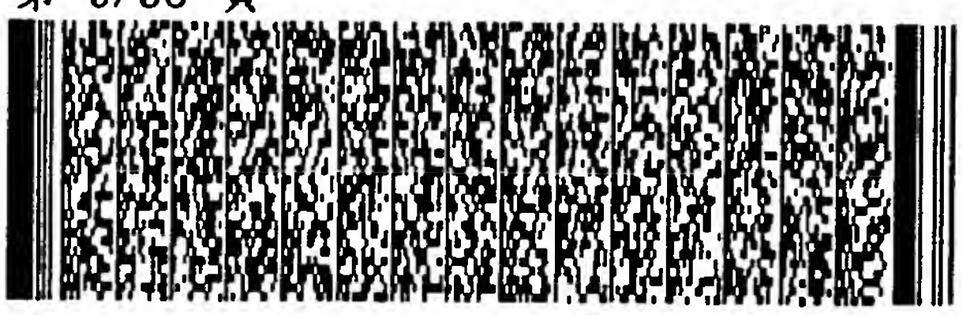


第 8/58 頁

第 9/58 頁

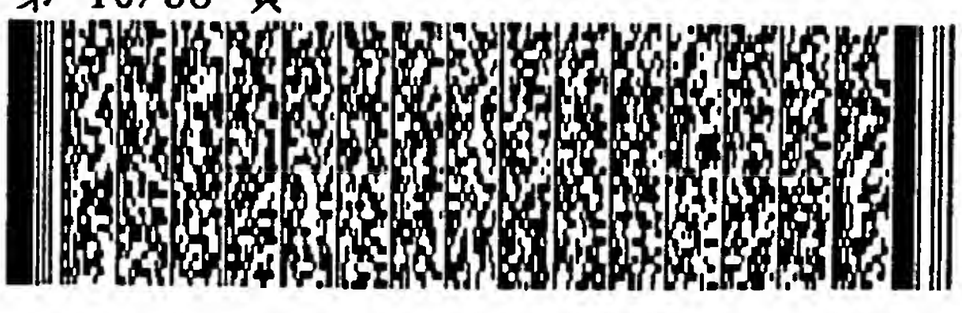


第 9/58 頁

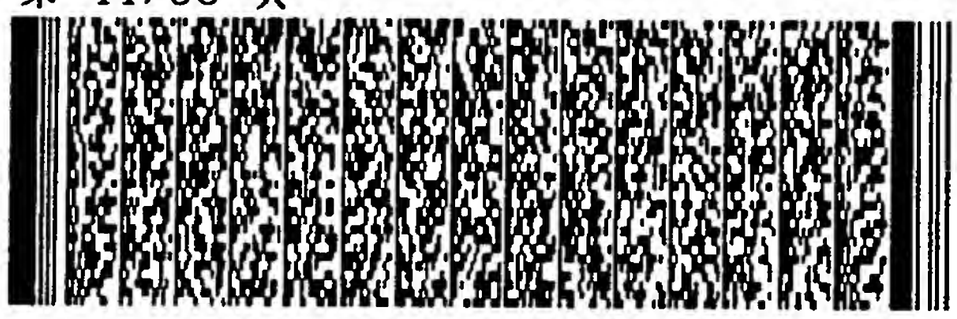


第 10/58 頁

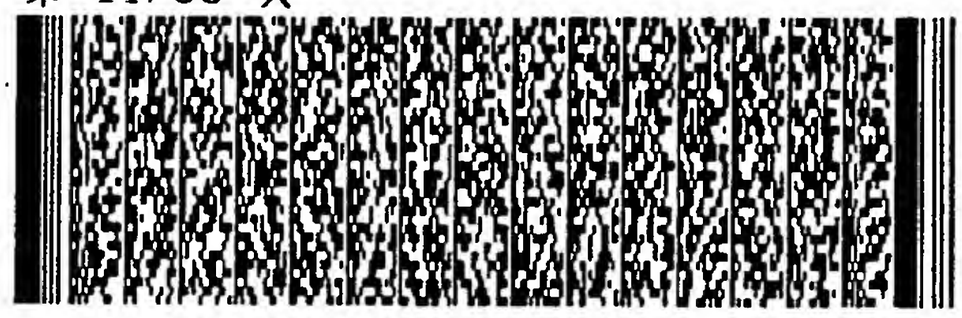
第 10/58 頁



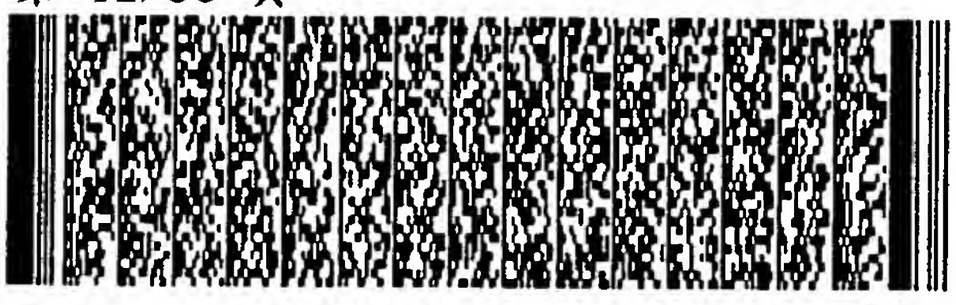
第 11/58 頁



第 11/58 頁

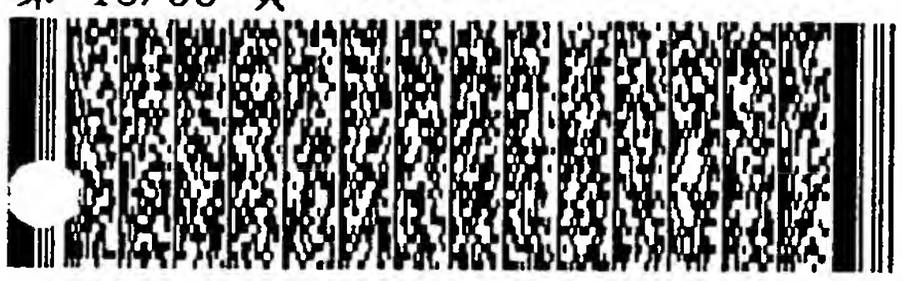


第 12/58 頁

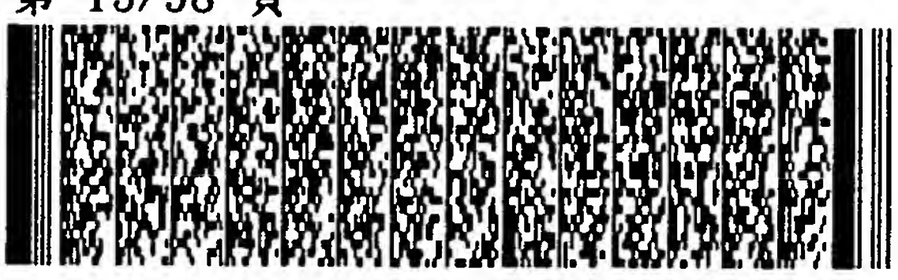


第 12/58 頁

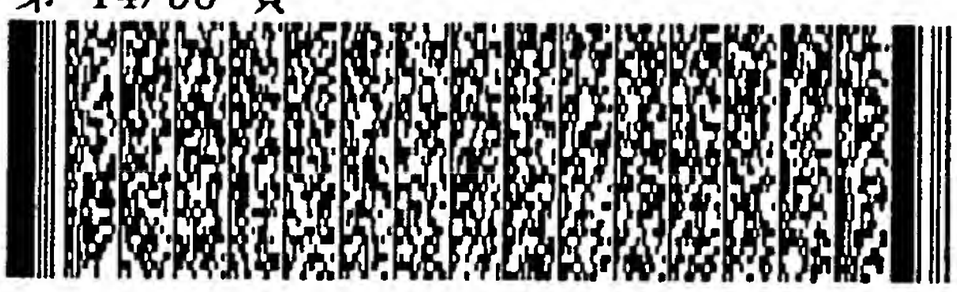
第 13/58 頁



第 13/58 頁

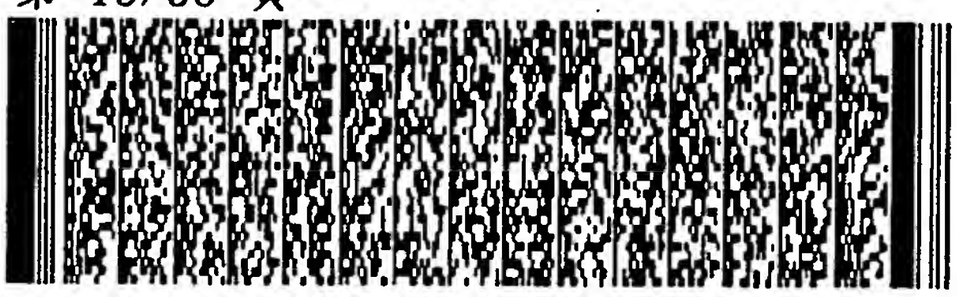


第 14/58 頁



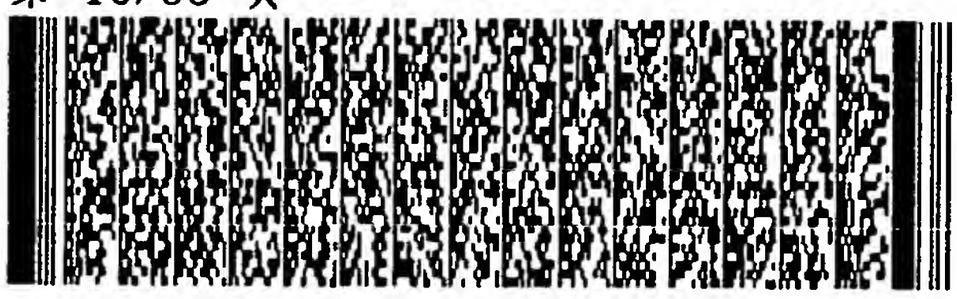
第 14/58 頁

第 15/58 頁

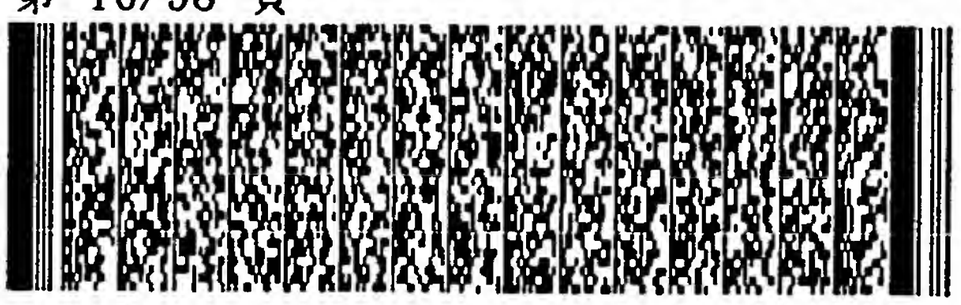


第 15/58 頁

第 16/58 頁

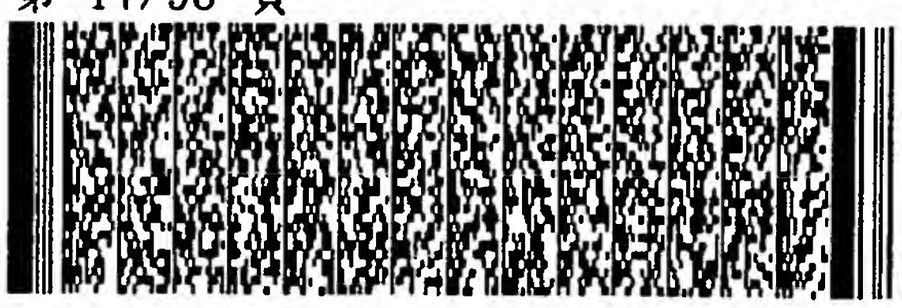


第 16/58 頁

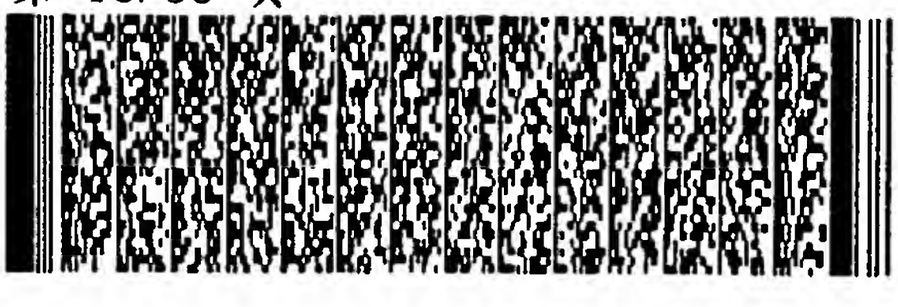


第 17/58 頁

第 17/58 頁

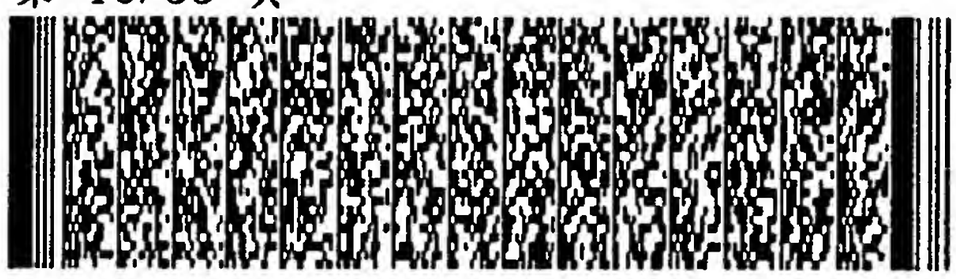


第 18/58 頁

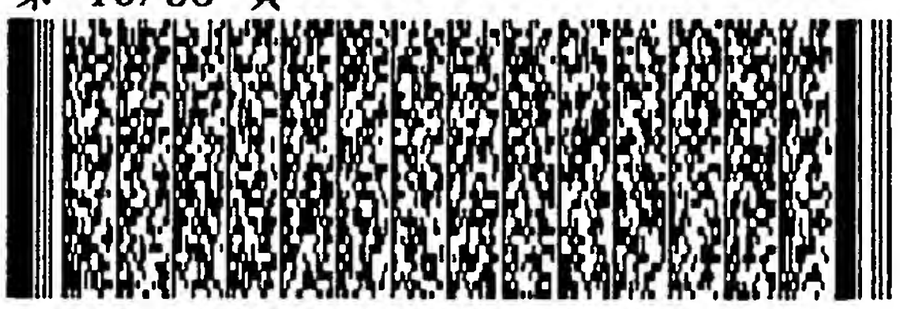


第 18/58 頁

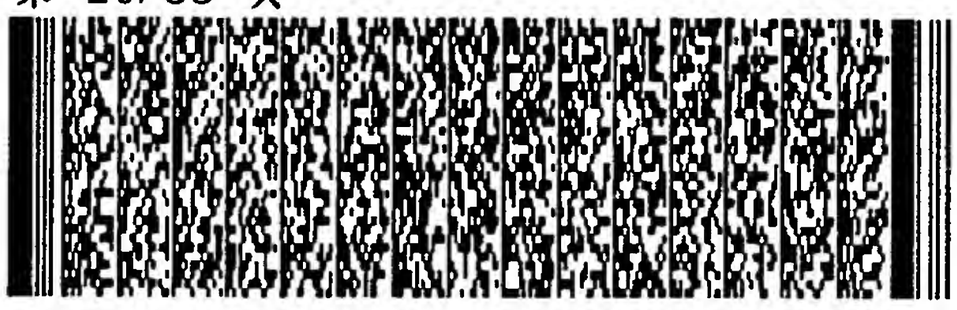
第 19/58 頁



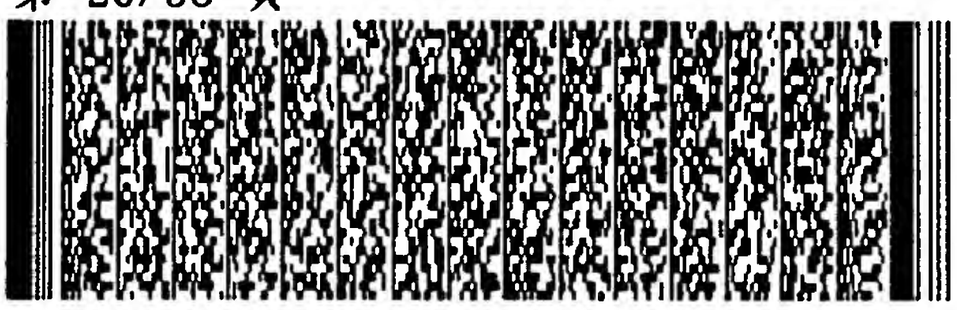
第 19/58 頁



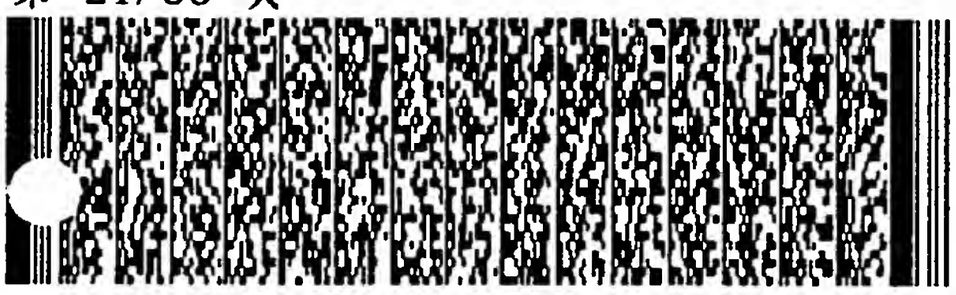
第 20/58 頁



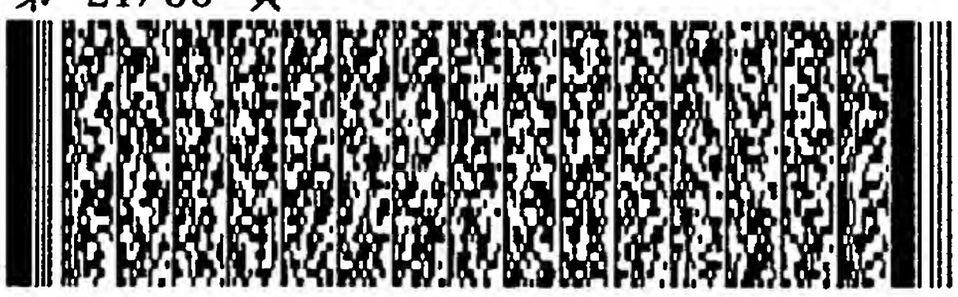
第 20/58 頁



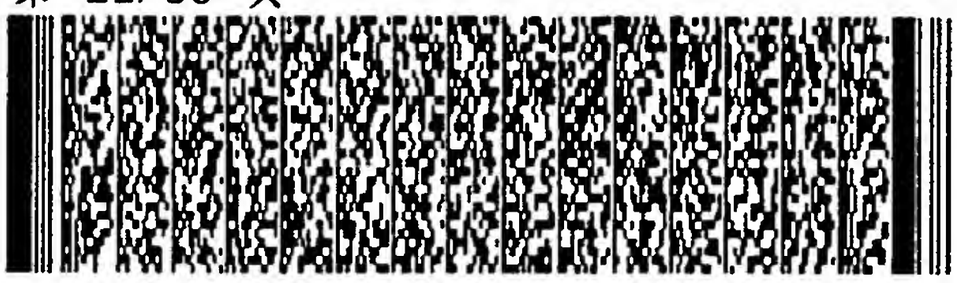
第 21/58 頁



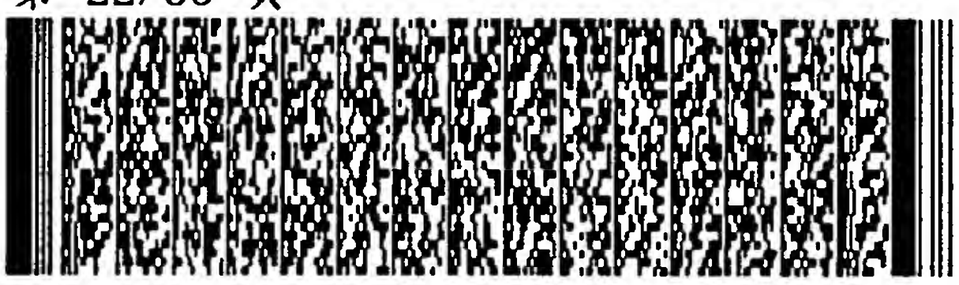
第 21/58 頁



第 22/58 頁

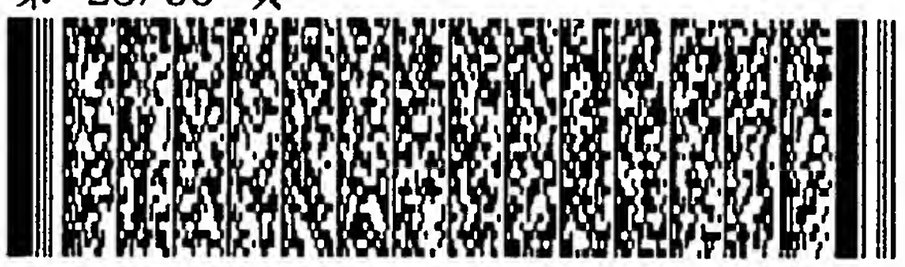


第 22/58 頁

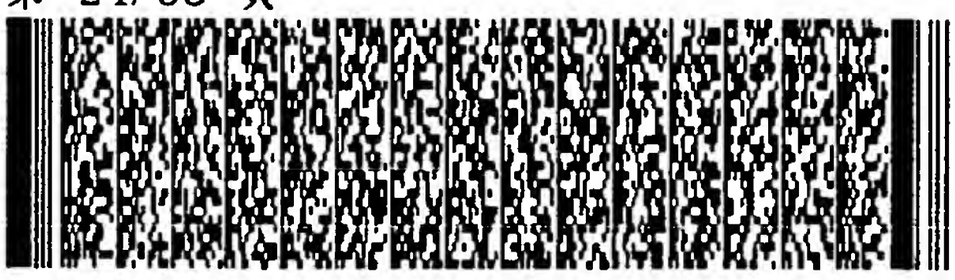


第 23/58 頁

第 23/58 頁

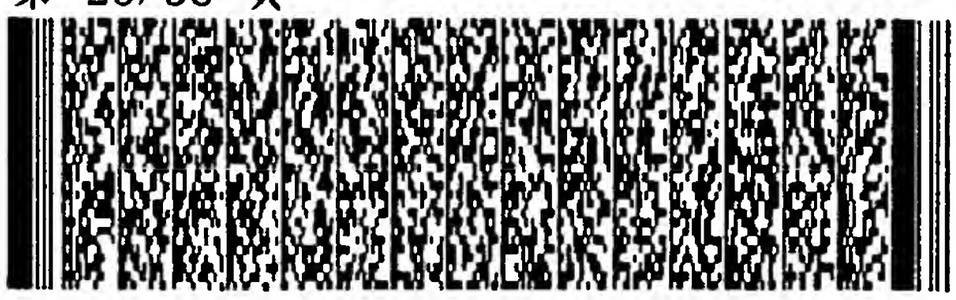


第 24/58 頁

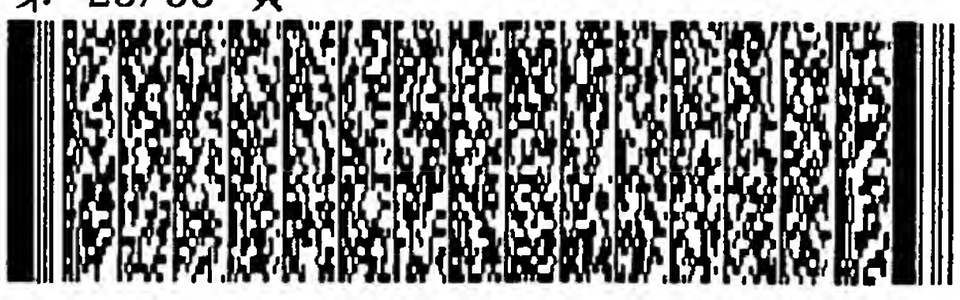


第 24/58 頁

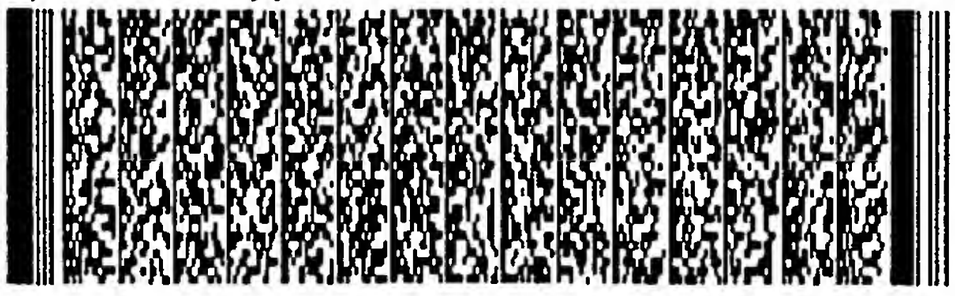
第 25/58 頁



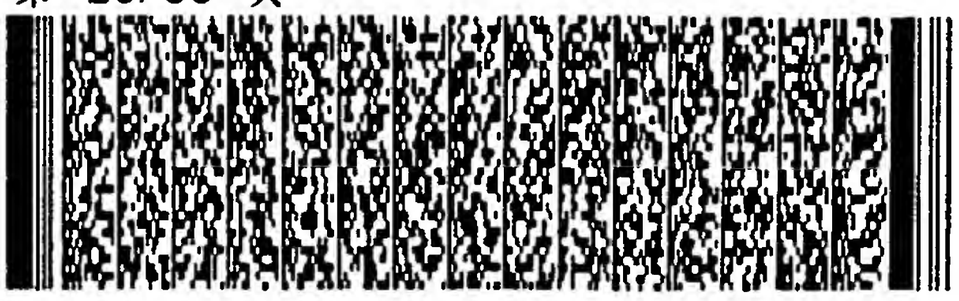
第 25/58 頁



第 26/58 頁

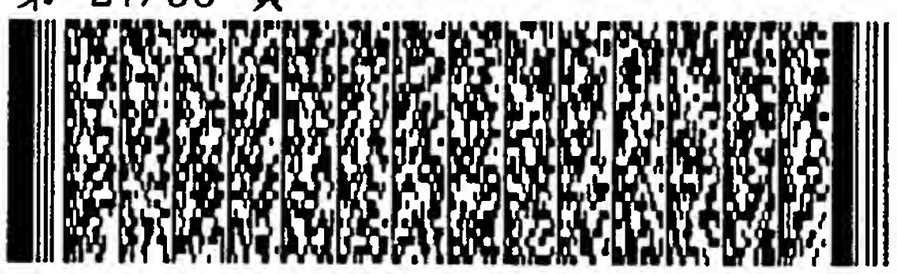


第 26/58 頁

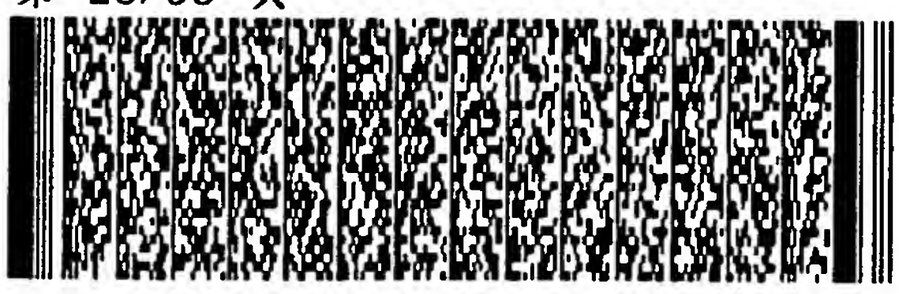


第 27/58 頁

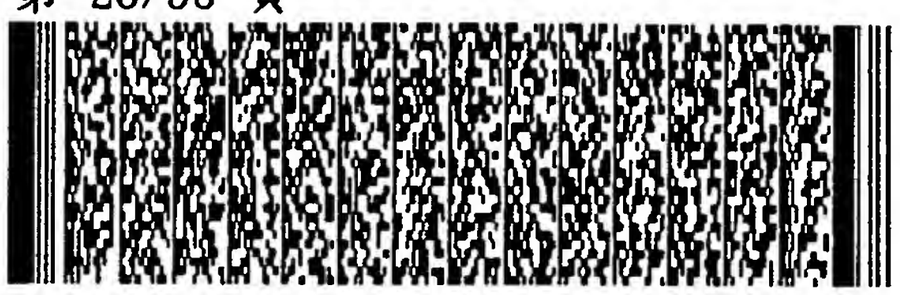
第 27/58 頁



第 28/58 頁

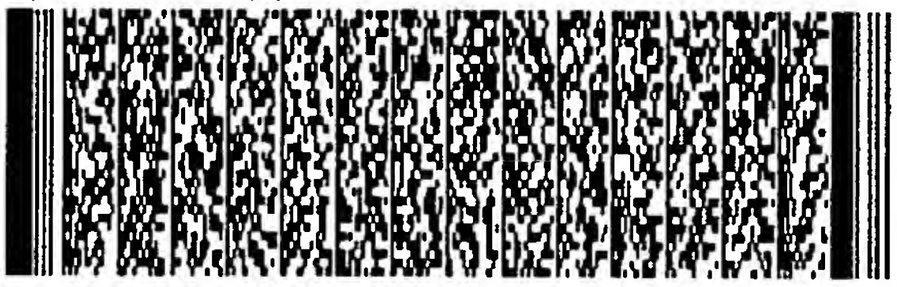


第 28/58 頁

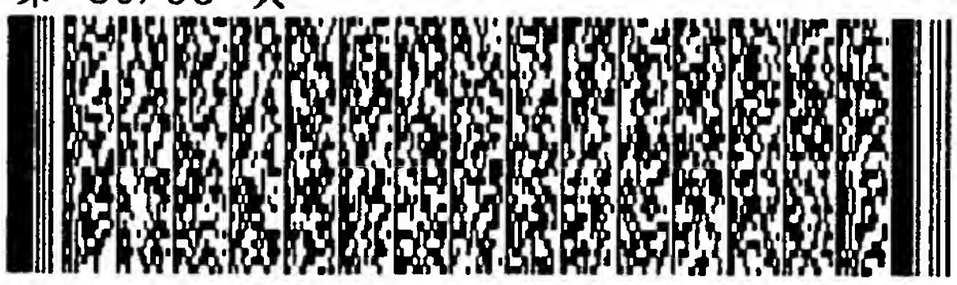


第 29/58 頁

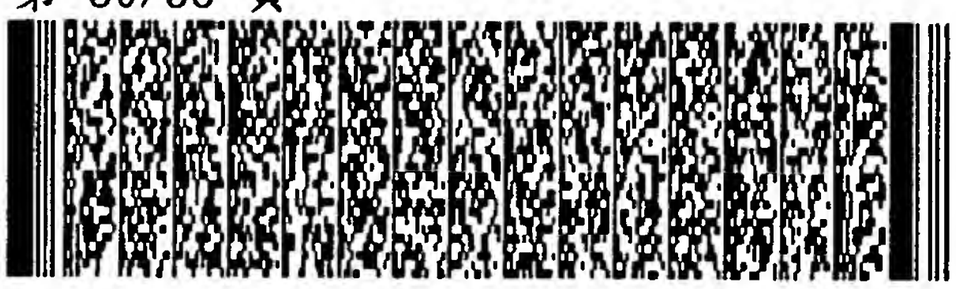
第 29/58 頁



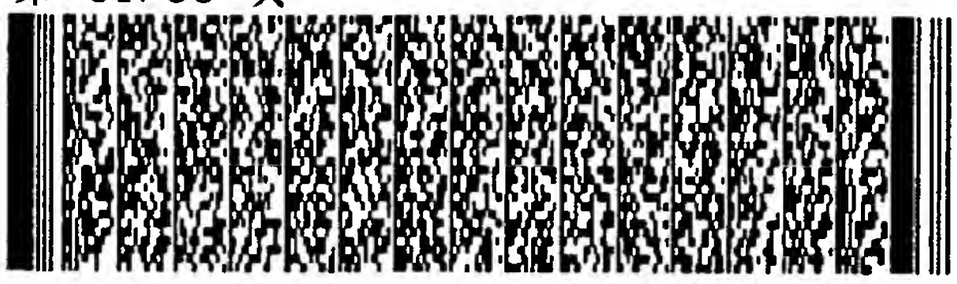
第 30/58 頁



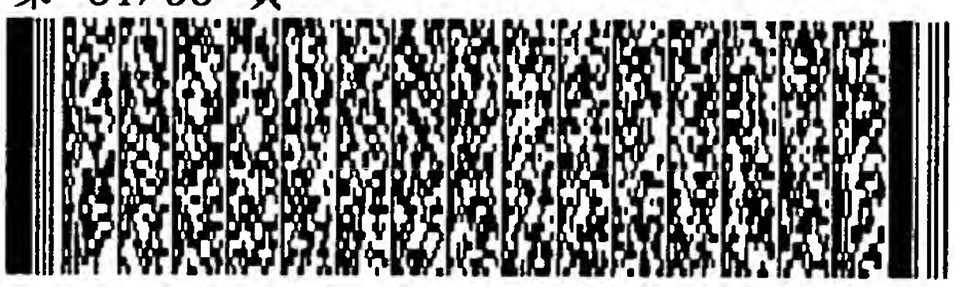
第 30/58 頁



第 31/58 頁



第 31/58 頁

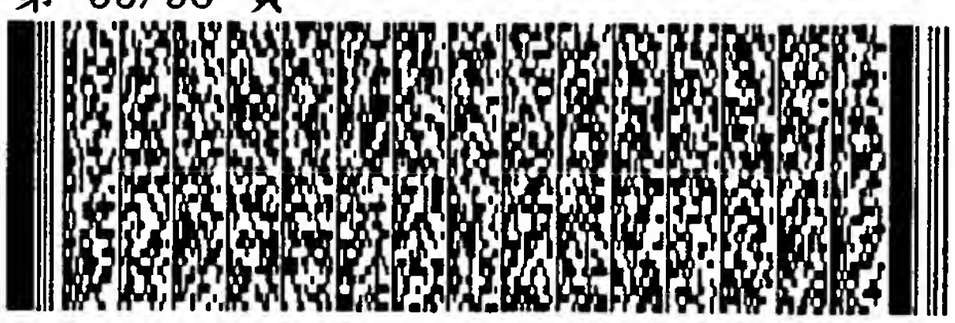


第 32/58 頁

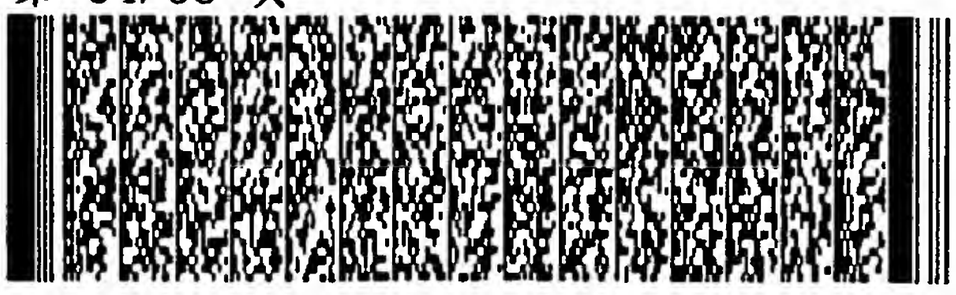
第 32/58 頁

第 33/58 頁

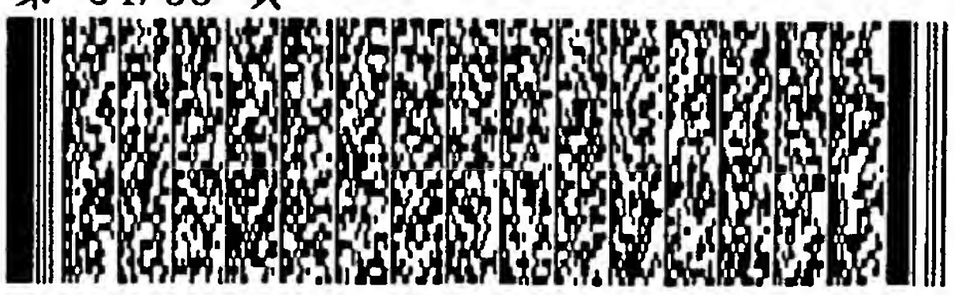
第 33/58 頁



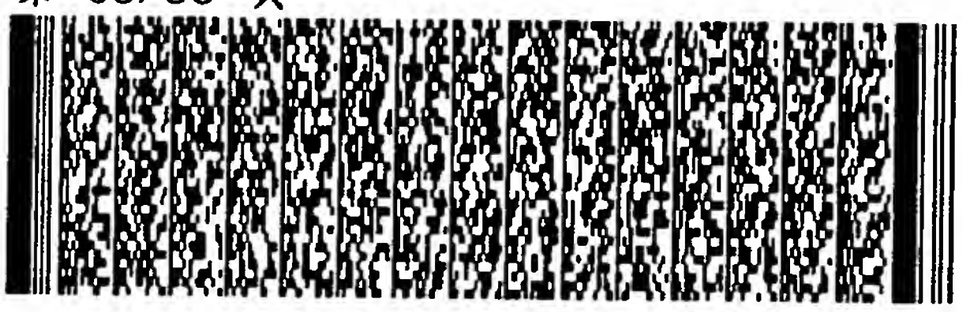
第 34/58 頁



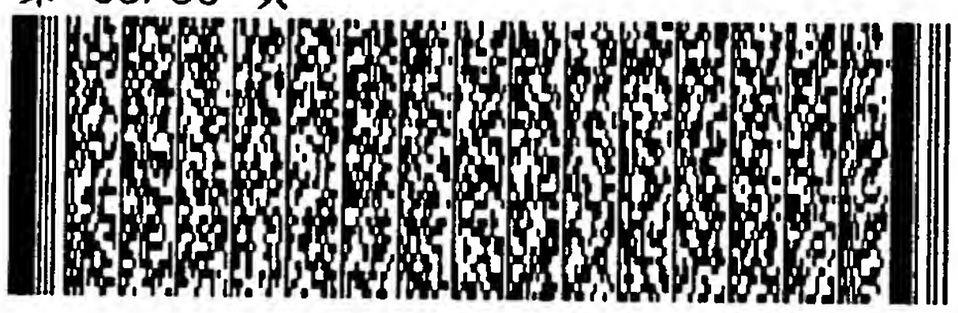
第 34/58 頁



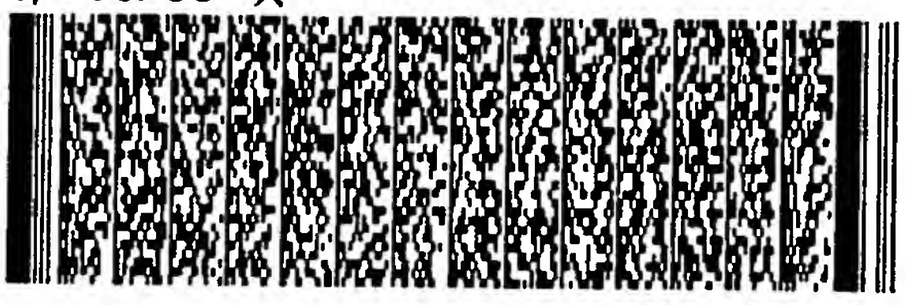
第 35/58 頁



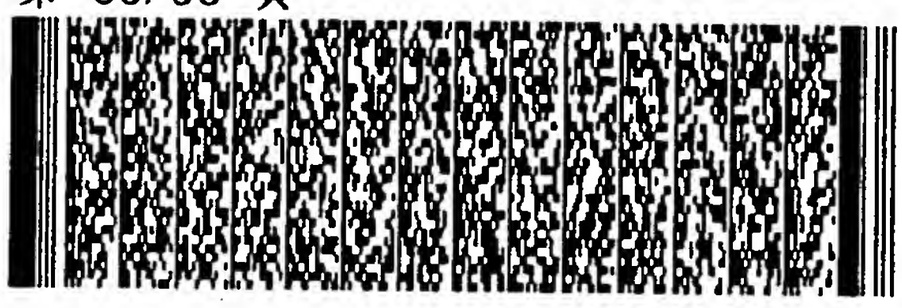
第 35/58 頁



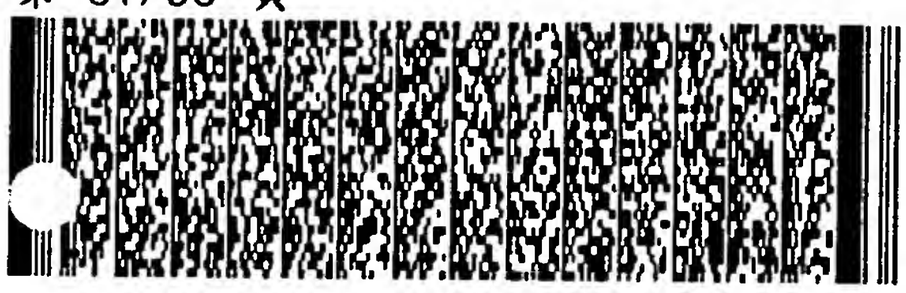
第 36/58 頁



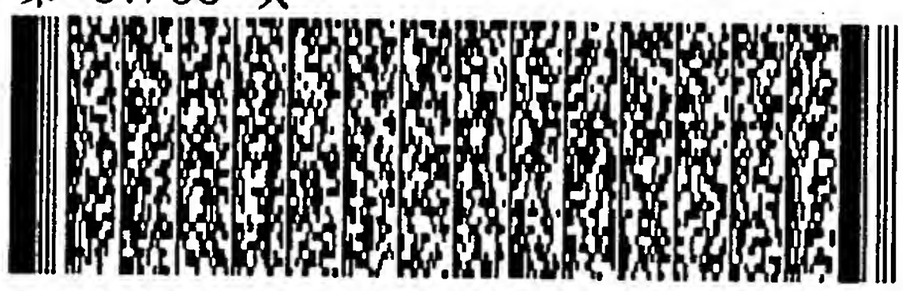
第 36/58 頁



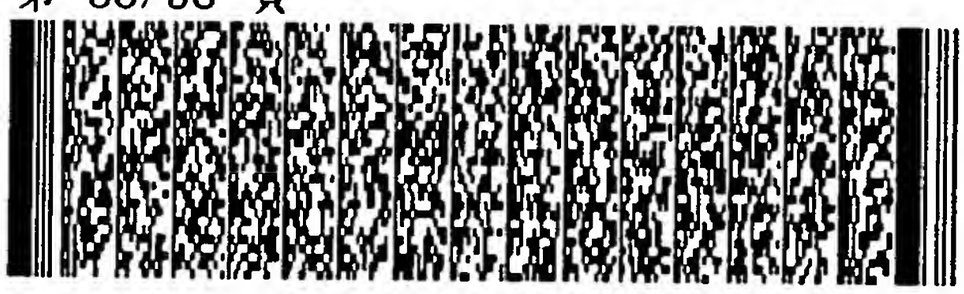
第 37/58 頁



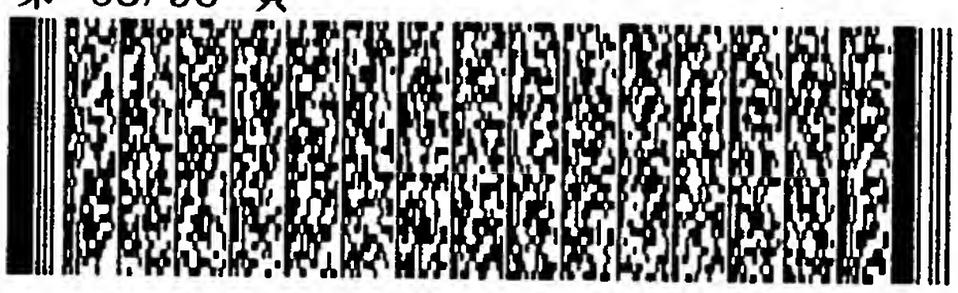
第 37/58 頁



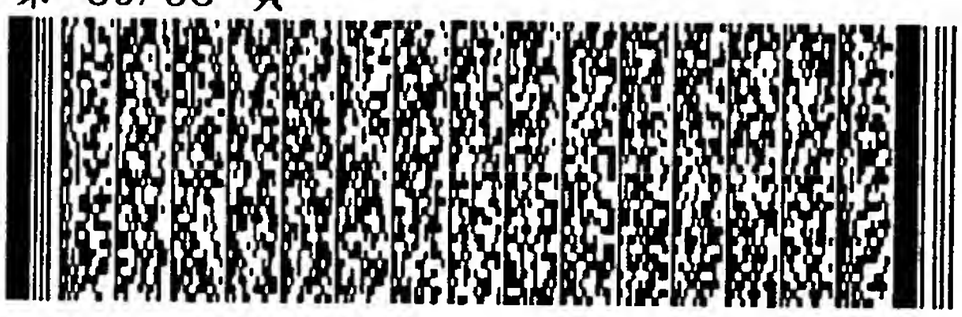
第 38/58 頁



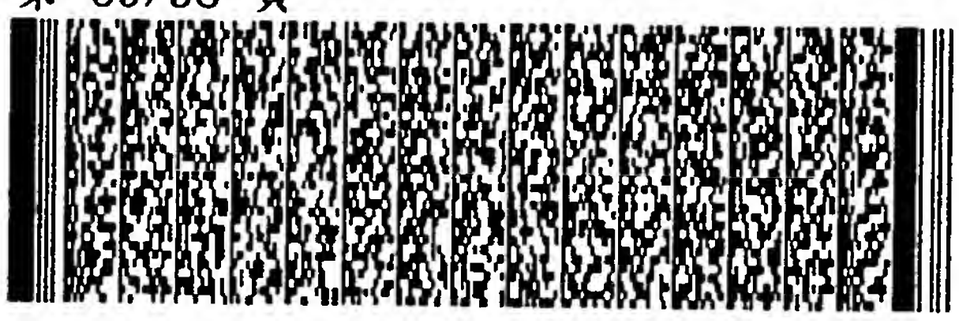
第 38/58 頁



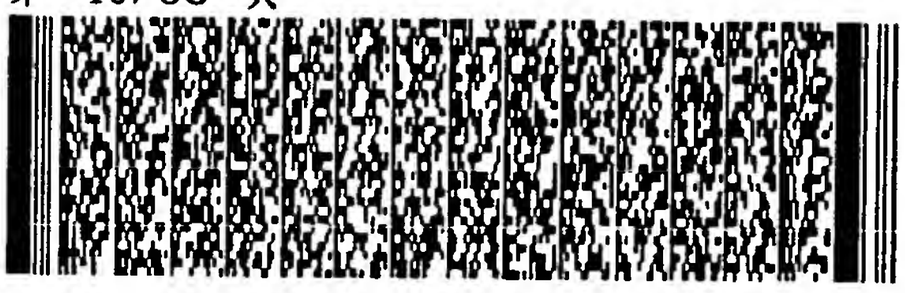
第 39/58 頁



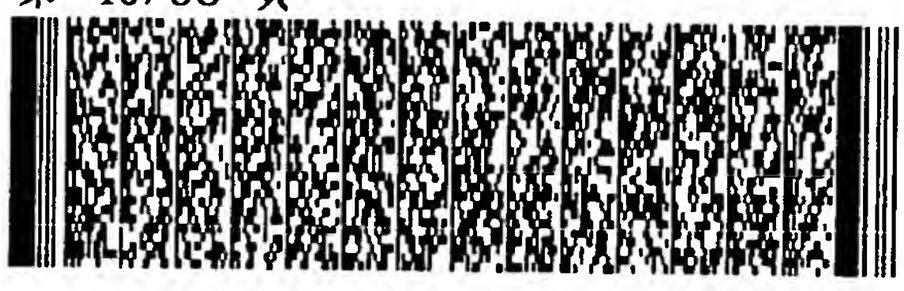
第 39/58 頁



第 40/58 頁

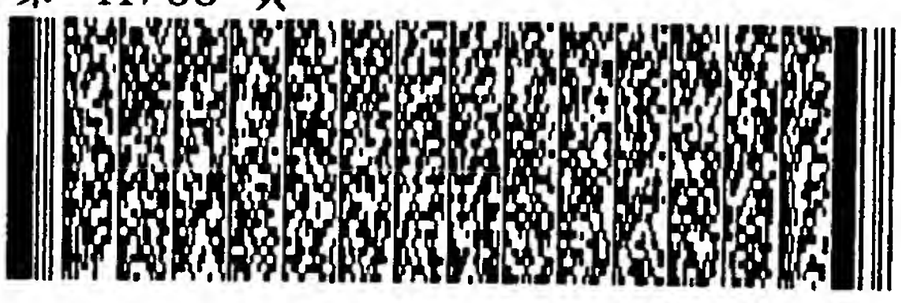


第 40/58 頁



第 41/58 頁

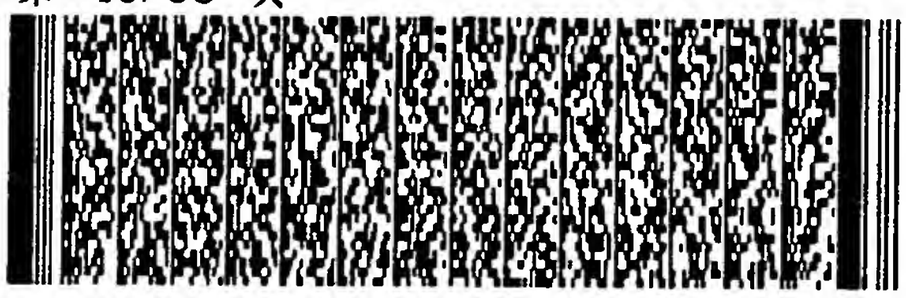
第 41/58 頁



第 42/58 頁

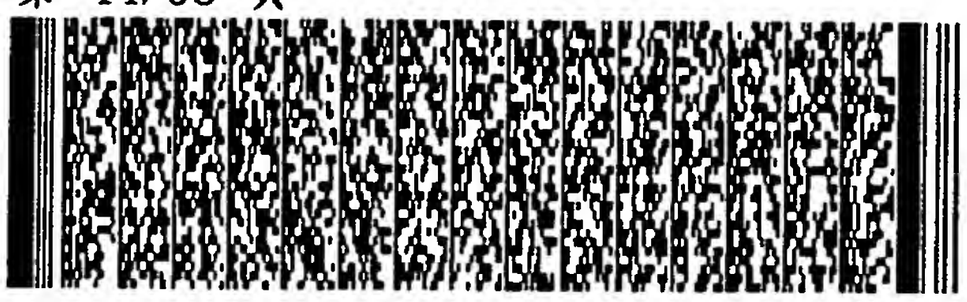
第 42/58 頁

第 43/58 頁



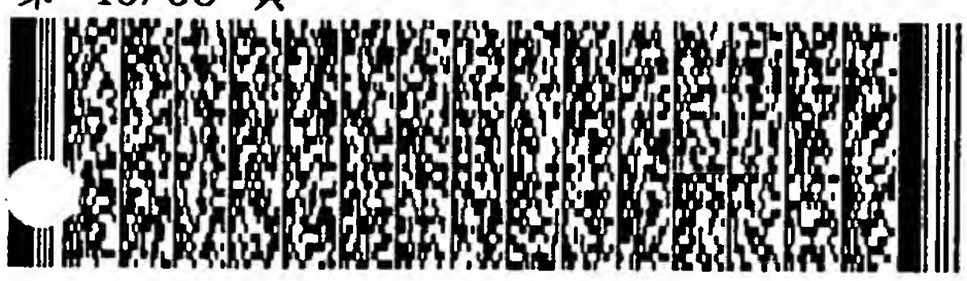
第 43/58 頁

第 44/58 頁

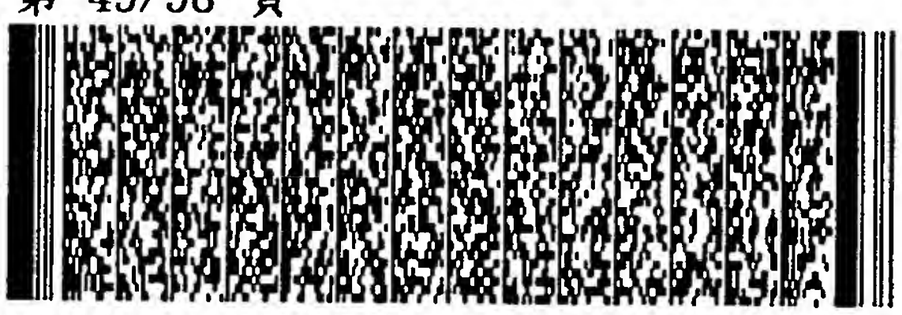


第 44/58 頁

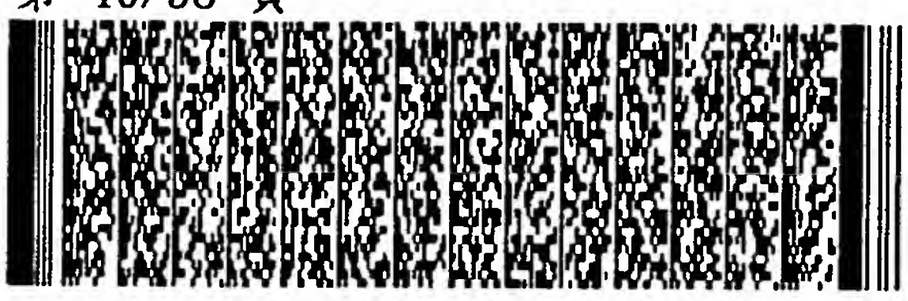
第 45/58 頁



第 45/58 頁

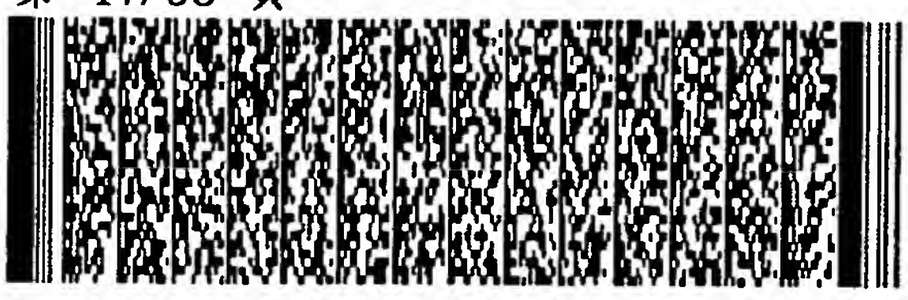


第 46/58 頁

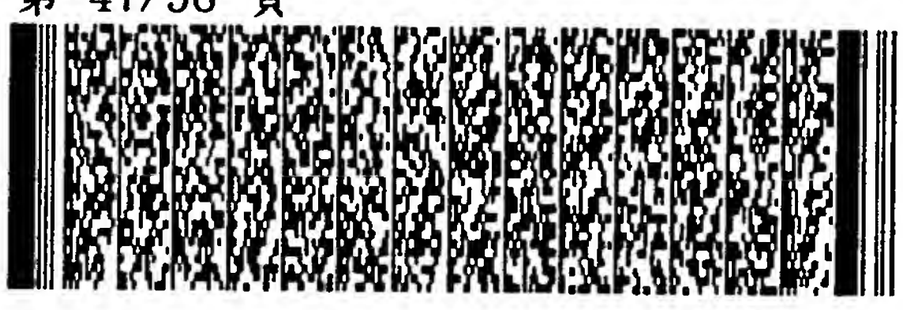


第 46/58 頁

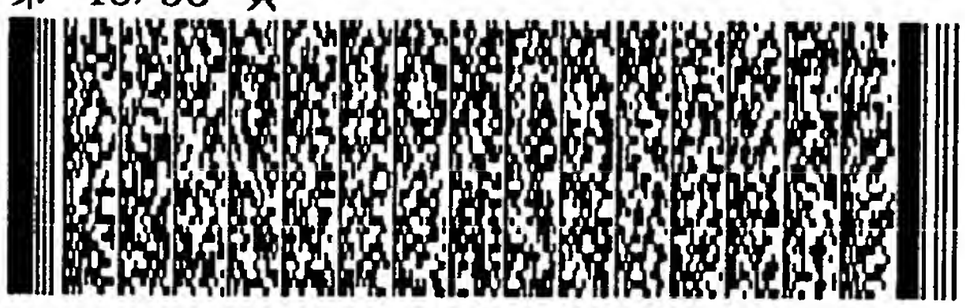
第 47/58 頁



第 47/58 頁

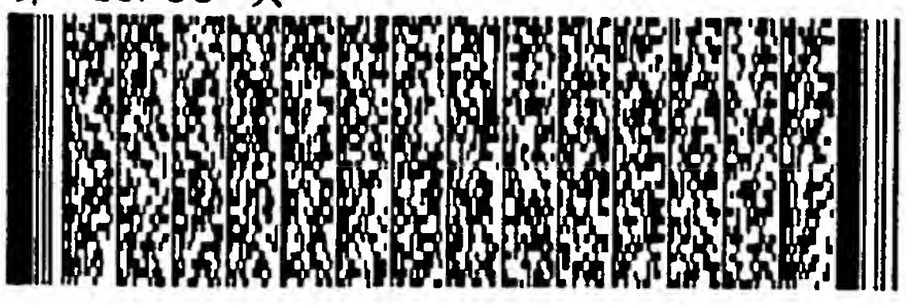


第 48/58 頁

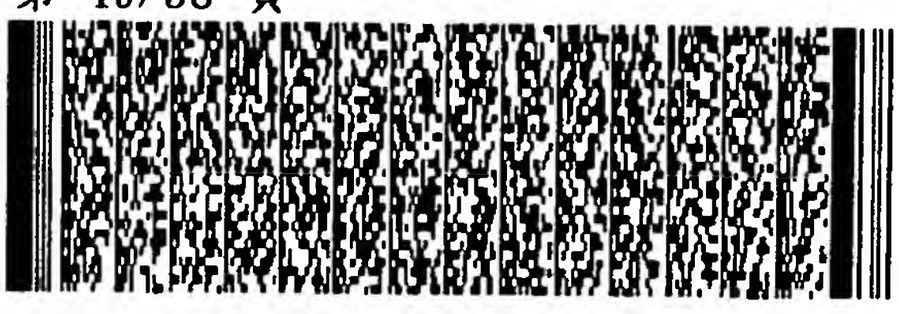


第 48/58 頁

第 49/58 頁

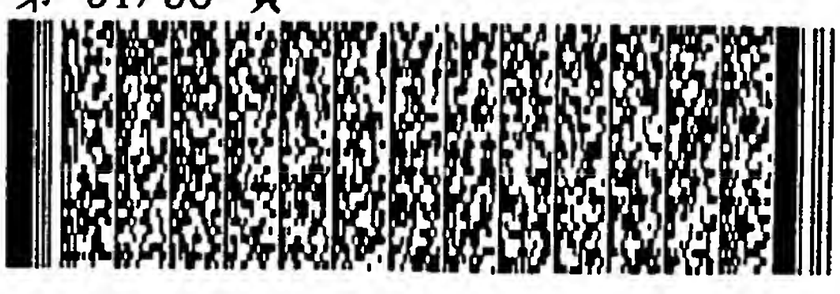


第 49/58 頁

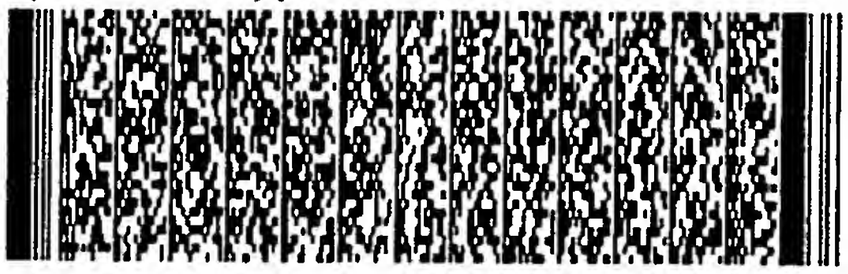


第 50/58 頁

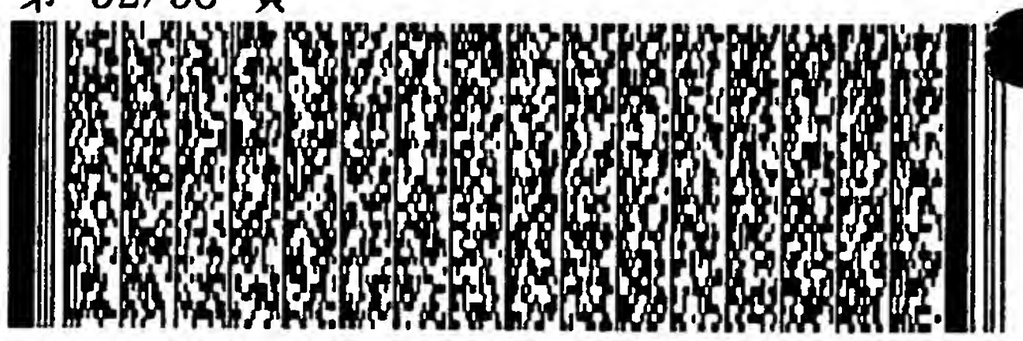
第 51/58 頁



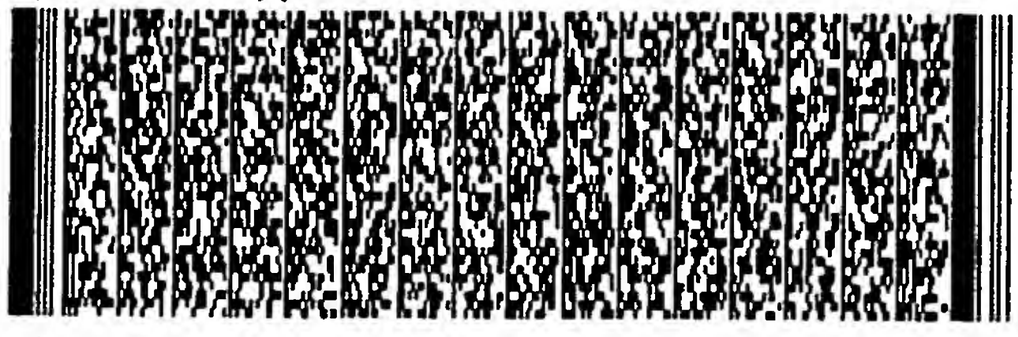
第 51/58 頁



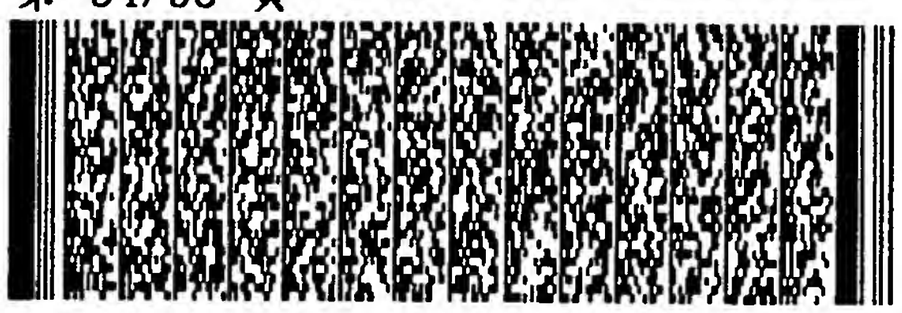
第 52/58 頁



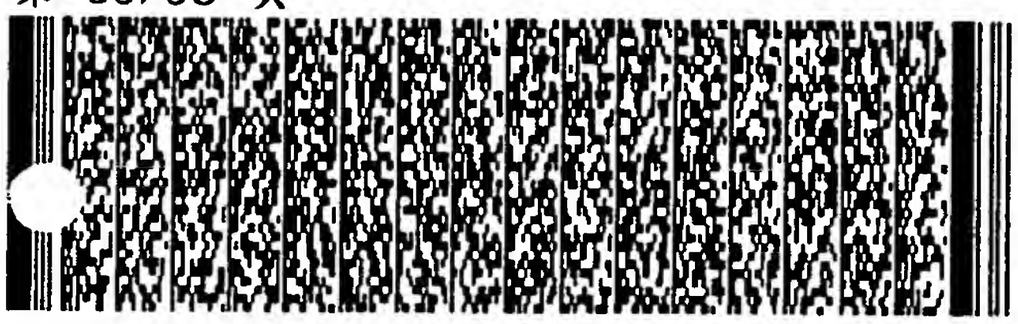
第 53/58 頁



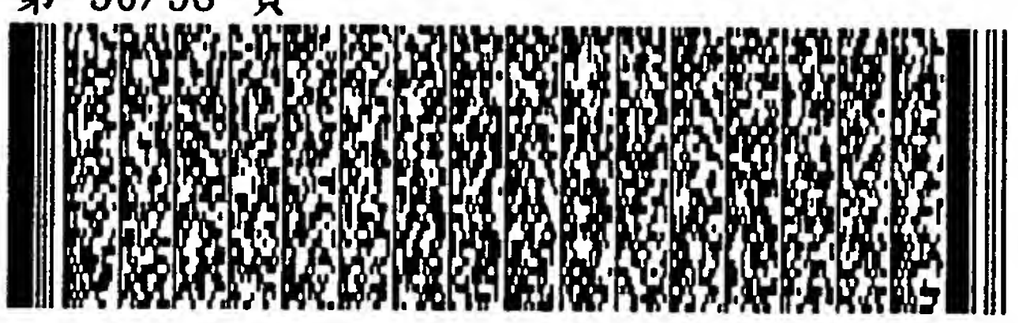
第 54/58 頁



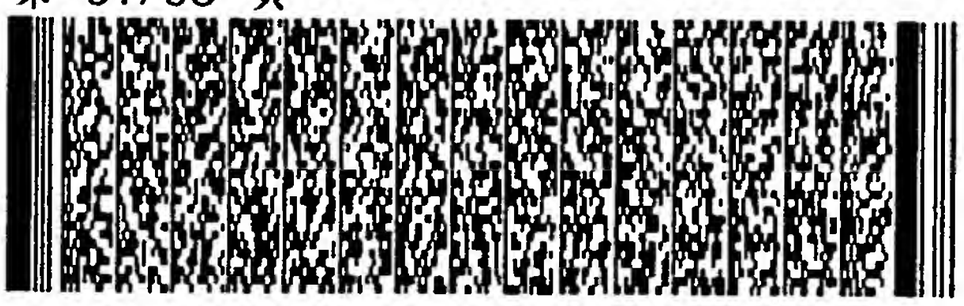
第 55/58 頁



第 56/58 頁



第 57/58 頁



第 58/58 頁